Predictive Task Assignment in Spatial Crowdsourcing: A Data-driven Approach

Yan Zhao
School of Computer Science and Technology
Soochow University
Suzhou, China
zhaoyan@suda.edu.cn

Kai Zheng*
University of Electronic Science and Technology of China
Chengdu, China
zhengkai@uestc.edu.cn

Yue Cui
University of Electronic Science and Technology of China
Chengdu, China
cuipaofu@gmail.com

Han Su
University of Electronic Science and Technology of China
Chengdu, China
hansu@uestc.edu.cn

Feida Zhu
Singapore Management University
Singapore
fdzhu@smu.edu.sg

Xiaofang Zhou
University of Queensland
Brisbane, Australia
zxf@itee.uq.edu.au

***Abstract*—With the rapid development of mobile networks and the widespread usage of mobile devices, spatial crowdsourcing, which refers to assigning location-based tasks to moving workers, has drawn increasing attention. One of the major issues in spatial crowdsourcing is task assignment, which allocates tasks to appropriate workers. However, existing works generally assume the static offline scenarios, where the spatio-temporal information of all the workers and tasks is determined and known a priori. Ignorance of the dynamic spatio-temporal distributions of workers and tasks can often lead to poor assignment results. In this work we study a novel spatial crowdsourcing problem, namely Predictive Task Assignment (PTA), which aims to maximize the number of assigned tasks by taking into account both current and future workers/tasks that enter the system dynamically with location unknown in advance. We propose a two-phase data-driven framework. The prediction phase hybrids different learning models to predict the locations and routes of future workers and designs a graph embedding approach to estimate the distribution of future tasks. In the assignment component, we propose both greedy algorithm for large-scale applications and optimal algorithm with graph partition based decomposition. Extensive experiments on two real datasets demonstrate the effectiveness of our framework.**

***Index Terms*—prediction, task assignment, spatial crowdsourcing**

I. Introduction

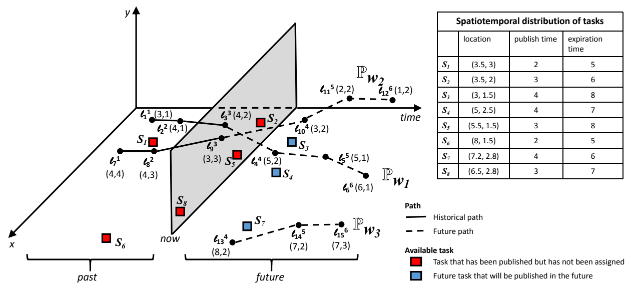
Along with the ubiquity of GPS-equipped smart devices and the high availability of wireless network, a new class of crowdsourcing that has enabled people to move as multi-modal sensors collecting and sharing various types of high-fidelity spatio-temporal data instantaneously, also known as Spatial Crowdsourcing (SC), has drawn increasing attention. Specifically, with spatial crowdsourcing, requesters can issue spatial tasks, such as taking photos/videos, monitoring traffic condition and reporting local hot spots, to the SC server dynamically and workers are assigned to these tasks by the server based on their locations and other constrains, which is referred to as *task assignment*.

There have been extensive studies on task assignment in SC, most of which are mainly based on the assumption of static offline scenarios, i.e., the locations of workers and tasks are known a priori either explicitly or implicitly. However, spatial crowdsourcing is a real-time platform in practical scenarios, on which workers and tasks become online dynamically with locations unknown in advance. Some recent work has explored the online assignment approaches in SC, where newly arrived tasks are assigned to the suitable workers based on the current task assignment [1]–[3]. Nevertheless they do not take into account future workers/tasks that have not entered the system.

Existing studies have shown that most people make journeys of a repetitive nature, such as going to and from a place of work, which makes predicting the location/route of a worker based on her previous traveling history possible [4]. In addition, by analyzing the task execution trajectories, along which a worker performs spatial tasks, we can not only understand individual's mobility patterns but also obtain valuable insights about her task execution behavior, which can be further utilized to improve the quality of spatial crowdsourcing. Peng et al. [5] are the first to utilize historical data to enhance the quality of task assignment by predicting the spatial distribution of workers/tasks in the next time instance. In this work we will go further in this direction and use a data-driven approach to predict locations/routes of workers and locations of tasks for a longer time duration, and then optimize the global task assignment based on this prediction.

Figure 1 illustrates an example of the dynamic spatial task assignment problem with three workers' paths indicated as $\mathbb{P}_{w_1}$, $\mathbb{P}_{w_2}$ and $\mathbb{P}_{w_3}$, and spatial tasks shown as $\{s_1, ..., s_8\}$. Each path is a sequence of locations with time stamps (i.e., location l_8^2 and l_9^3 of path $\mathbb{P}_{w_2}$ with time stamp 2 and 3 respectively), and the current time instance is 3. Each worker is associated with her reachable distance range, which is set

* Corresponding author: Kai Zheng.



Spatiotemporal distribution of tasks			
	location	publish time	expiration time
S_1	(3.5, 3)	2	5
S_2	(3.5, 2)	3	6
S_3	(3, 1.5)	4	8
S_4	(5, 2.5)	4	7
S_5	(5.5, 1.5)	3	8
S_6	(8, 1.5)	2	5
S_7	(7.2, 2.8)	4	6
S_8	(6.5, 2.8)	3	7

🟥 Task that has been published but has not been assigned
🟦 Future task that will be published in the future

Fig. 1. Running Example

to 1.2. In addition, each spatial task, published and expired at different time instances, is labelled with its location where it will be performed only once. The online spatial crowdsourcing problem in our work is to assign tasks to the suitable workers at both current and future timestamps so as to maximize the total number of assigned tasks. To better understand the spatio-temporal distributions of workers and tasks, we map all the location points of the 3D space (in Figure 1) into a 2D spatial plane, as shown in Figure 2.

It is intuitive to assign the nearby tasks to workers without violating the spatio-temporal constraint of workers and tasks to maximize the current assignment at every instance of time, referred to as Maximum Task Assignment (MTA) instance problem [6]. Therefore, in our example, we assign task s_1 to worker w_2, and s_2 to w_1 at the present time to achieve the maximal number of assigned tasks, i.e., 2. Similarly, in the next time instance (i.e., 4), we can assign w_1 with s_3 and w_3 with s_6, achieving the maximal number of assigned tasks at time instance 4, i.e., the maximal number equals to 2. However, the remaining tasks cannot be assigned to workers since workers are not able to arrive at the locations of the remaining tasks after performing their own assigned tasks. As a result, such an assignment strategy during a time period (e.g., time instance 3-6) leads to the overall number of assigned tasks, $4\ (= 2 + 2)$, which is depicted in Figure 2(a).

However, the above assignment approach just tries to maximize the current assignment (i.e., local optimization instead of global optimization) without considering the future workers/tasks that may dynamically appear in the future time instances. When future workers/tasks are known a priori, the task assignment problem can be reduced to the classic Maximum Coverage Problem [7] and its variants. Nevertheless, the main challenge with SC comes from the dynamism of the arriving workers/tasks, which renders an optimal solution infeasible in the online scenario.

To overcome this challenge, we propose a data-driven framework, called Data-driven Predictive Spatial Task Assignment (DPSTA), which consists of prediction phase and assignment phase. The first phase aims to predict the spatio-temporal distributions of workers/tasks in the future time instances. For worker prediction, based on two different task assignment (e.g., location/route-specific task assignment) strategies, we introduce the Spatial Temporal Recurrent Neural Network (ST-RNN) model to predict the appearance location for each future worker, and design a hybrid model to predict the potential route for each current/future worker based on her traveling history. For task prediction, we design a Path Constrained DeepWalk (PC-DeepWalk) algorithm to estimate the number of future tasks, and then utilize the Kernel Density Estimation (KDE) approach to predict the locations for future tasks by considering the tasks as spatial point events. In the second phase, the Location/Route-specific Maximal Valid Task Sets (L/R-MaxVTSs, see Definition 6) are firstly calculated for each worker based on the current and future workers/tasks. Then we need to tackle the computation issue in the huge search space when enumerating all possible combinations of the valid task sets of each worker, which increases exponentially with respect to the number of workers. For the sake of efficiency, we propose a greedy algorithm that tries to assign each worker with the maximal L/R-MaxVTSs from the unassigned tasks. We also develop an exact graph partition based decomposition algorithm that finds the optimal assignment result in terms of the total number of assigned tasks. Figure 2(b) illustrates the task scheduling and assignment by applying our exact location-specific method that covers 6 tasks, and Figure 2(c) shows the task assignment result by applying our exact route-specific method, which covers 7 tasks.

The contributions of this paper can be summarized as follows:

i) We provide a Data-driven Predictive Spatial Task Assignment (DPSTA) framework for spatial crowdsourcing with the aim of optimizing the global task assignment when both workers and tasks appear dynamically in a given time duration.

ii) Two novel strategies are proposed to predict the future locations and routes for workers based on their traveling histories.

iii) We design an effective graph embedding mechanism to estimate the spatio-temporal distributions for tasks.

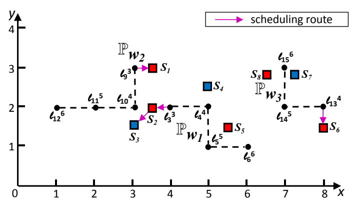
iv) We propose greedy and optimal algorithms for task assignment to trade off assignment efficiency and effectiveness.

v) Extensive experiments are conducted with real-world datasets, where the empirical results confirm that our solutions are effective in assigning spatial tasks in a real-time manner.

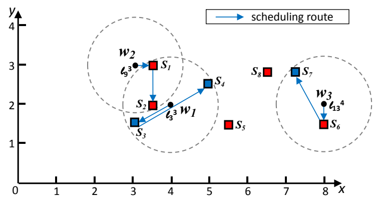
The remainder of this paper is organized as follows. Section II introduces the related work and Section III provides notations and the proposed problem, along with a brief introduction of the framework overview. In Section IV and V, we design different prediction strategies for workers and tasks respectively. The greedy and exact task assignment algorithms are then presented in Section VI, followed by the experimental results in Section VII. Finally, we conclude the paper in Section VIII.

II. Related Work

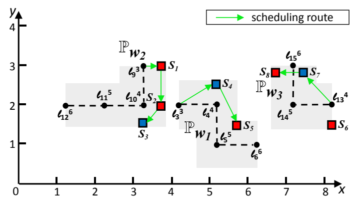
Spatial Crowdsourcing (SC) can be deemed as one of the main enablers to complete location-based tasks [8]–[11]. According to the task publish mode, SC can be classified into two categories namely *Server Assigned Tasks* (SAT) mode and *Worker Selected Tasks* (WST) mode. Most existing works adopt SAT mode, where the SC server takes charge of the task assignment process. In SAT mode, the server assigns



(a) Snapshot Routes by MTA



(b) Online Routes by Our Location-specific Method



(c) Online Routes by Our Route-specific Method

Fig. 2. Task Scheduling and Assignment

proper tasks to nearby workers in order to achieve some system optimization goals such as maximizing the number of assigned tasks after collecting all the locations of workers/tasks [5], [12], [13], or maximizing the coverage of required skills of workers [14]. A majority of the research carried out so far has been based on the assumption of static offline scenarios, i.e., the spatio-temporal distributions of workers and tasks are known a priori. It means that these studies do not consider the challenges of real-time spatial task assignment, where workers and tasks can come and go at any time.

However, SC is a real-time platform, through which both workers and tasks occur dynamically. Recent studies focus on devising algorithms to solve the online task assignment problems in SC [1]–[3], [15], [16]. Specifically, in [1], the online spatial task assignment problem is addressed by the Online Minimum Bipartite Matching approach, where only tasks are released dynamically following the random order model. By considering the dynamic workers and tasks, [2] focuses on online mobile micro-task allocation, which assumes the spatial distributions of workers/tasks are known and their arrival orders follow the random order model. However, the occurrences of workers/tasks are affected by complicated factors that are hard to be captured by a single fixed model. The online route planning problem for a crowd worker is also proposed to maximize the number of completed tasks [16], in which only the occurrence of tasks is dynamic. In order to assign three types of objects (e.g., workers, tasks and workplaces) that dynamically appear, Song et al. [3] design a Trichromatic

TABLE I
SUMMARY OF NOTATIONS

Notation	Definition
s	Spatial task
$s.r$	Released time of spatial task s
$s.l$	Location of spatial task s
$s.e$	Expiration time of spatial task s
$s.c$	Category of spatial task s
t	A time instance
T	Time instance set
w	Available worker
$w.\phi$	Available time of worker w
$w.range$	Reachable distance of worker w
$w.\mathbb{P}$	Route of worker w during her available time
S_w	A task set for w
R	A task sequence
$t(l)$	Arrival time of particular location l
$c(a,b)$	Travel time from a to b
$L\text{-}VTS(w)$	A location-specific valid task set of w
$R\text{-}VTS(w)$	A route-specific valid task set of w
$L\text{-}MaxVTS(w)$	A location-specific maximal valid task set of w
$R\text{-}MaxVTS(w)$	A route-specific maximal valid task set of w
A	A spatial task assignment
$\mathbb{A}$	A spatial task assignment set

Online Matching model to maximize the total utility of worker-task-workplace matching. The aforementioned works only consider the current and newly released workers/tasks but ignore the future ones.

The closest related research to ours is [5], which studies the prediction-based online spatial task assignment problem. However, it differs from our work in terms of the problem setting and objectives. First, [5] assigns tasks by worker-task matching based on the spatio-temporal distributions in the current and next time instance, while we assign tasks by giving a scheduled task sequence for each worker on the basis of the spatio-temporal distributions in the current and multiple future time instances. Second, the goal in [5] is to maximize the overall quality score of assignments under the given traveling cost budget constraints, whereas we aim to maximize the total number of assigned tasks in the given time duration. It is particularly noticeable that we compare the task assignment results between [5] and our work in the experiments.

III. Problem Statement

In this section, we briefly introduce a set of preliminary concepts in the context of self-incentivised single task assignment in spatial crowdsourcing with SAT mode, and then give an overview of our framework. Table I lists the major notations used throughout the paper.

A. Preliminary Concepts

Definition 1 (Spatial Task): A spatial task, denoted by $s =< s.r, s.l, s.e, s.c >$, is a task released at time $s.r$, to be performed at location $s.l$, and will expire at $s.e$ ($s.r \leq s.e$), where $s.l : (x, y)$ is a point in the 2D space. Each task s is also labelled with a category $s.c$.

For simplicity and without loss of generality, we assume: 1) single task assignment mode, i.e., the server assigns each task to one worker only; 2) the processing time of each task is 0, which means that a worker will go to the next task upon finishing the current task. However, our proposed techniques are not restricted to the above assumptions.

Definition 2 (Available Worker): Given a set of time instances, $T = \{t, t+1, ..., t+n\}$ (t is the current time instance), an available worker, $w =< w.\phi, w.range, w.\mathbb{P} >$, is associated with her available time instances $w.\phi = \{t+k, t+k+1, ..., t+g\}$ ($\subset T$), reachable distance $w.range$, and the corresponding traveling route $w.\mathbb{P}$, which consists of a set of time-stamped locations (i.e., $w.\mathbb{P} = (w.l_1^{t+k}, w.l_2^{t+k+1}, ..., w.l_{|w.\phi|}^{t+g})$).

Definition 3 (Task Sequence): Given a worker w and a set of tasks assigned to her S_w, a task sequence on S_w, denoted as $R(S_w) = (s_1, s_2, ..., s_{|S_w|})$, represents the order by which w visits each task in S_w. The arrival time of w at task $s_i \in S_w$ (i.e., the time of completing task s_i) can be computed as follows:

$$t_{w,R}(s_i.l) = \begin{cases} t_{w,R}(s_{i-1}.l) + c(s_{i-1}.l, s_i.l) & \text{if } i \neq 1 \\ t_{now} + c(w.l, s_1.l) & \text{if } i = 1, \end{cases}$$

where $c(a, b)$ is the travel time from location a to location b, t_{now} is the current time, and $w.l$ denotes the starting location, from which w begins to accept the task assignment. When the context of w and R is clear, we use $t(s_i.l)$ to denote $t_{w,R}(s_i.l)$.

Definition 4 (Location-specific Valid Task Set): Given a time instance set T and a worker w's starting location $w.l$, a task set S_w is called a Location-specific Valid Task Set (L-VTS) for w, if there exists a task sequence $R(S_w) = (s_1, s_2, ..., s_{|S_w|})$, such that $\forall s_i \in S_w$:

i) $t(s_i.l) \leq s_i.e$, and

ii) $t(s_i.l) \in w.\phi$ ($\subset T$), and

iii) $d(w.l, s_i.l) \leq w.range$, where $d(a, b)$ is a given distance between location a and b.

Definition 5 (Route-specific Valid Task Set): Given a time instance set T and a route $w.\mathbb{P} = (w.l_1, w.l_2, ..., w.l_{|w.\phi|})$ for worker w, a task set S_w is called a Route-specific Valid Task Set (R-VTS) for w, if there exists a task sequence $R(S_w) = (s_1, s_2, ..., s_{|S_w|})$, such that $\forall s_i \in S_w$:

i) $t(s_i.l) \leq s_i.e$, and

ii) $t(s_i.l) \in w.\phi$ ($\subset T$), and

iii) $D(s_i.l, w.\mathbb{P}) \leq \frac{w.range}{2}$, where $D(a, b)$ is a given distance of location a from route b.

Definition 6 (Location/Route-specific Maximal VTS): Given a set of time instances T, a Location/Route-specific Valid Task Set S_w is maximal if none of its super sets is still valid for a worker w, which is called Location/Route-specific Maximal Valid Task Set (L/R-MaxVTS).

Definition 7 (Spatial Task Assignment): Given a set of time instances $T = \{t, t+1, ..., t+n\}$, a set of workers and tasks available during T, a spatial task assignment, denoted by A, consists of a set of $< worker, VTS >$ pairs in the form of $< w_1, VTS(w_1) >, < w_2, VTS(w_2) >, ...$

Let $A.S$ denote the set of tasks that are assigned to all workers, i.e., $A.S = \cup_{w \in W} S_w$, and $\mathbb{A}$ denote all possible ways of assignments. The problem investigated in our paper can be formally stated as follows.

Problem Statement: Given a set of time instances $T = \{t, t+1, ..., t+n\}$ (t is the current time instance), the Predictive Task Assignment (PTA) problem aims to find the global optimal assignment A_{opti}, such that $\forall\ A_i \in \mathbb{A}$, $|A_i.S| \leq |A_{opti}.S|$.

B. Framework Overview

The main novelty of our proposed DPSTA framework is that the server will take into account the workers and tasks at not only the current time instance but also the next consecutive time instances. Therefore, an immediate challenge is to get an accurate estimation of future distributions for both workers and tasks in spatio-temporal dimensions. To this end, in this paper we propose a novel spatial crowdsourcing framework comprising of two components: worker/task prediction and task assignment, as illustrated by Figure 3.

The first component aims to predict the future spatio-temporal distributions of workers/tasks from the task execution trajectory history of workers and the release history of tasks. We propose different strategies for worker and task prediction respectively. Specifically, by considering the task execution history of a worker as sequential data, we utilize Sequential Pattern Mining (SPM) method to mine the frequent time instances when she is more likely to complete tasks as her available time. Then we propose two strategies of spatial distribution prediction for each worker in her available time: 1) Spatial Temporal Recurrent Neural Network (ST-RNN) based location prediction; 2) a hybrid model based route prediction. As for task prediction, since spatial tasks can be regarded as spatial point events, a Path Constraint DeepWalk (PC-DeepWalk) model is designed to obtain the number of future tasks at each time instance, and then the Kernel Density Estimation (KDE) approach is employed to predict the location distributions of tasks in the future time instances.

The second component needs to assign the tasks to the suitable workers by scheduling a task sequence for each worker in order to achieve the maximal task assignment. We first calculate the whole MaxVTSs for every worker, and then the subsequent task assignment has to tackle the computational issue in the huge search space when enumerating all possible combinations of the MaxVTSs of each worker. We propose both Greedy Task Assignment (GTA) algorithm that tries to assign each worker with the maximal MaxVTS from the unassigned tasks and Optimal Task Assignment (OTA) algorithm that obtains the global optimal task assignment.

IV. Worker Prediction

SC involves mobility in the physical world, influenced by a range of location-dependent factors, among which workers' behavioral patterns play a key role for task assignment. In this part, we first utilize a frequent mining approach to detect the available time for workers. Then the ST-RNN model and a hybrid model (integrating pattern matching strategy and spatio-temporal sequential correlation) are employed to find the future locations and routes where a worker tends to perform tasks during her available time.

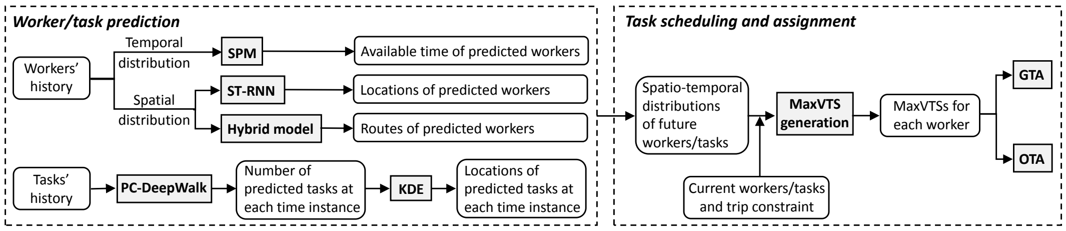

Worker/task prediction
Workers' history
Temporal distribution
Spatial distribution
SPM
ST-RNN
Hybrid model
Available time of predicted workers
Locations of predicted workers
Routes of predicted workers
Tasks' history
PC-DeepWalk
Number of predicted tasks at each time instance
KDE
Locations of predicted tasks at each time instance
Task scheduling and assignment
Spatio-temporal distributions of future workers/tasks
MaxVTS generation
MaxVTSs for each worker
GTA
OTA
Current workers/tasks and trip constraint

Fig. 3. DPSTA Framework Overview

A. Temporal Distribution Prediction

Considering the time-ordered task execution events as time-series data, we mine the frequent time instances when a worker is more likely to perform tasks from worker's task execution history by using a frequent pattern mining algorithm, which has been studied extensively in time-series databases [17]. Let $T_k^w = \{t_1, t_2, ..., t_m\}$ be a set of time-ordered time instances, at which worker w performs tasks in the k-th day, and $T^w = \{T_1^w, T_2^w, ..., T_n^w\}$ denote all the time instances of n days for worker w in her task execution history. A set of time instances, of which the occurrence frequencies exceed a given minimum support threshold, can be extracted by scanning T^w, and each continuous time instance set is called *available time* (i.e., $w.\phi$) for worker w.

B. Spatial Distribution Prediction

Once the available time is obtained for each worker, we propose two strategies to predict spatial distributions of future workers. We assume each worker has a GPS device (e.g., a GPS-enabled mobile phone) keeping track of her positions.

1) Location Prediction: Inspired by the success of ST-RNN [18] model for finding the sequential correlations among POIs, we apply it to predict the location of each future worker at the beginning of her available time, from which the worker tends to perform tasks.

In ST-RNN, given a set of potential workers W and a set of locations L, $\mathbf{p}_w \in \mathbb{R}^d$ and $\mathbf{q}_l \in \mathbb{R}^d$ are the latent vectors of worker w and location l respectively. Each location is associated with its coordinate and each worker w has a set of historical locations where she just passed by, i.e., $L^w = \{l_{t_1}^w, l_{t_2}^w, ...\}$. The architecture of ST-RNN model is shown in Figure 4, which contains three layers: input layer, hidden layer, and output layer. The input layer contains the latent vector of the location worker w visits at time t_i, i.e., $\mathbf{q}_{l_{t_i}^w} \in \mathbb{R}^d$. The hidden layer is the key component of ST-RNN, in which the vector representation of hidden layers for w at current time instance t can be computed as follows:

$$\mathbf{h}_{t,l_t^w}^w = f\Big(\sum_{l_{t_i}^w \in L^w, t-\hat{v}<t_i<t} \mathbf{S}_{l_t^w - l_{t_i}^w} \mathbf{T}_{t-t_i} \mathbf{q}_{l_{t_i}^w} + \mathbf{C}\mathbf{h}_{t-\hat{v}, l_{t-\hat{v}}^w}^w\Big),$$

where $\mathbf{h}_{t,l_t^w}^w$ is the representation of worker w at time t, v is the width of time window, $\mathbf{S}_{l_t^w - l_{t_i}^w}$ is the distance-specific transition matrix for geographical distance between l_t^w and $l_{t_i}^w$, $\mathbf{T}_{t-t_i}$ denotes the time-specific transition matrix for time interval (e.g., $t - t_i$), and $\mathbf{C}$ is the recurrent connection of the previous status propagating sequential signals. The activation function

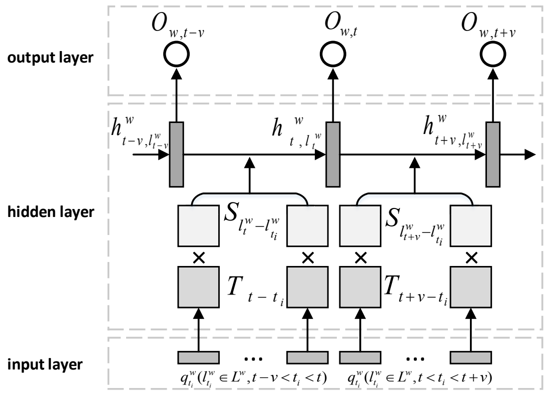


Fig. 4. Diagram of ST-RNN Model

$f(x)$ is chosen as a $sigmod$ function $f(x) = exp(1/1+e^{-x})$. Note that the location l_{t-v}^w may not exist in the history L^w, thus we use the approximate value $\hat{v}$, the most closed value to v, as the local window width to guarantee $l_{t-\hat{v}}^w$ to be contained in the history, i.e., $l_{t-\hat{v}}^w \in L^w$. Finally, the prediction of ST-RNN can be yielded via calculating inner product of worker and location representations in the output layer. The prediction of whether worker w would go to location l at time t can be computed as:

$$O_{w,t,l} = (\mathbf{h}_{t,l}^w + \mathbf{p}_w)^T \mathbf{q}_l, \quad (1)$$

where $\mathbf{h}_{t,l}^w$ captures her dynamic interests under the specific spatial and temporal contexts and $\mathbf{p}_w$ is the permanent representation of worker w indicating her interest and activity range. To accurately estimate the location at the beginning of worker's available time, ST-RNN partitions time intervals and geographical distances into discrete bins respectively, in which it learns the transition matrices for the upper/lower bound of the corresponding bins and calculates the transition matrices for other time intervals/geographical distances by linear interpolation. Bayesian Personalized Ranking (BPR) [19] and Back Propagation Through Time (BPTT) [20] are applied to learn the parameters (i.e., $\mathbf{S}$, $\mathbf{T}$, $\mathbf{C}$, $\mathbf{p}$, $\mathbf{q}$) in ST-RNN model.

2) Route Prediction: Since most individuals' daily outdoor movements are constrained by physical roads [21], we aim to predict the potential routes of all workers based on the GPS observations of workers' past trips in a road network. It has been proved to be an effective way to predict a moving object's future route based on the route pattern extracted from its historical trajectory data [4]. However, Pattern Matching Approach (PMA) suffers from sparsity problem, i.e., the available historical trajectories are far from being able to cover all possible trajectories, which may return no prediction results. To tackle this issue, we employ ST-RNN model into the prediction process when encountering no-pattern matching. By this way, the route prediction accuracy and robustness can be improved.

The overview of this hybrid model for a worker's route prediction is illustrated in Figure 5. The model first extracts the road corners from massive historical trajectories by a Characteristic Point-based Road Corner Extraction (CP-RCE) method [21], and then both the historical trajectory data and the query trajectory to be predicted are represented by the trajectory mapping approach based on these road corners to generate road corner-centric routes. During the process of route prediction, Pattern Matching Approach (PMA) is used to predict the future route on basis of the discovered frequent movement patterns. When encountering no-pattern matching, the ST-RNN model can be employed to predict the next moving road corner. Note that we simply use the worker's latest velocity as her future velocity to calculate the distance she is able to travel during her available time. Towards this distance, the given query route grows gradually based on the hybrid prediction model. In the following, we will elaborate the related technologies.

Road Corner Detection and Trajectory Mapping. Since the topology information of physical roads (e.g., road corners) is embedded in personal GPS trajectories, we detect road corners from these trajectories and utilize them to represent each trajectory. In particular, we employ the CP-RCE method, in which a set of characteristic points (i.e., GPS points where the trajectory's direction changes significantly) are generated by a linear fitting approach, and then the road corners are identified based on a Multiple Density Level Density-Based Spatial Clustering of Applications with Noise (MDL-DBSCAN) [21] algorithm. In this way, popular road corners that occur frequently in historical trajectories can be detected and trajectories can be abstracted by a set of corner-centric routes using these road corners accordingly.

Pattern Matching Approach (PMA). In the previous step, each trajectory is converted into an ordered sequence of road corners. Subsequently, we employ the well-known prefix-projected sequential pattern mining (PrefixSpan) [22] algorithm to discover the hidden moving frequent patterns from historical trajectories. PrefixSpan is a recursive algorithm, which finds the frequent prefix sequences first, then projects them into the projected databases and finds the frequent suffixes to concatenate with the prefix to get the frequent sequential pattern without generating candidate sequences. FP-tree, an indexing structure, is used to store the projected databases for efficiency.

Upon getting the frequent moving patterns, we implement the pattern matching procedure to find the candidate patterns whose prefix can match the road corner-centric route generated from the query trajectory by the longest last matching strategy [23]. The longest last matching strategy focuses on the relative matching coverage of the query route with respect to the frequent patterns to be matched in order to find a longest pattern as the predicted route.

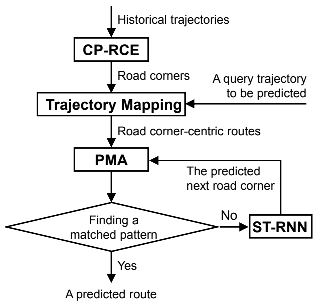

Historical trajectories
CP-RCE
Road corners
A query trajectory to be predicted
Trajectory Mapping
Road corner-centric routes
PMA
The predicted next road corner
Finding a matched pattern
No
ST-RNN
Yes
A predicted route

Fig. 5. Diagram of Route Prediction Model

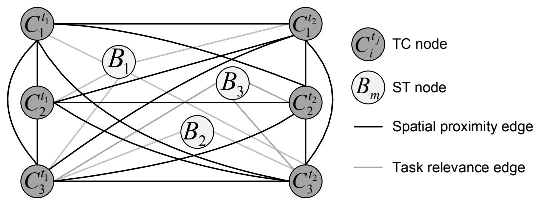


Fig. 6. Spatial Task-based Network

V. Task Prediction

It is crucial to understand where, when and what type of the tasks will be published in the future for a better global task assignment. Since spatial tasks have to be answered at specified locations, we consider the tasks as spatial point events. The planar Kernel Density Estimation (KDE) has been used widely for spatial point event analysis and detection [24], which aims to produce a smooth density surface of point events over space by computing event intensity as density estimation.

In this section, we employ the planar KDE approach to compute the density of spatial tasks to predict their locations by partitioning the study area into disjoint and uniform grids (e.g., 20×20). Before predicting tasks' locations, we need to estimate the number of potential tasks (with different types) that may fall into each grid cell in the future time instances. Instead of considering only temporal correlation of task counts in each cell [5], we construct a spatial task-based network and apply a network embedding method to predict task counts by taking spatial and temporal relationship into account in each cell in the future time instances.

A. *Task Number Prediction*

In this part, we predict the number of tasks for each grid in a set of future timestamps by using historical data. We first construct a network based on the spatial-temporal information of tasks, called spatial task-based network, $G = (V, E)$ (see Figure 6), in which V includes two types of nodes (i.e., temporal cell-based nodes and spatial task-based nodes) and E contains two types of edges (i.e., spatial proximity edges and task relevance edges). Apparently, the proposed spatial task-based network is a Heterogeneous Information Network (HIN) since its nodes and edges belong to multiple types. The nodes and edges are defined as follows:

Definition 8 (Temporal Cell-based (TC) Node): A temporal cell-based node, denoted as $C_i^{t_j}$, represents a specific spatial cell C_i of the grid at timestamp t_j.

Definition 9 (Spatial Task-based (ST) Node): A spatial task-based node, denoted as B_m, is a node representing a specific task type m.

Definition 10 (Spatial Proximity Edge): A spatial proximity edge, $e(C_i^{t_j}, C_k^{t_g})$, connects two TC nodes, representing the Spatial Proximity Relation (SPR) of TC nodes. The weight of the edge, denoted by $\mathcal{W}(C_i^{t_j}, C_k^{t_g})$, has a negatively correlation with the spatial distance between $C_i^{t_j}$ and $C_k^{t_g}$.

Definition 11 (Task Relevance Edge): A task relevance edge, $e(C_i^{t_j}, B_m)$, connects a TC node and a ST node, representing the Task Relevance Relation (TRR), i.e., the tasks (with a specific type m) are published in the spatial cell C_i in timestamp t_j. The weight of the edge, denoted as $\mathcal{W}(C_i^{t_j}, B_m)$, is the number of tasks (with a specific type m) that are published in spatial cell C_i in timestamp t_j.

In order to encode each node into a low dimensional vector and maintain the structural information, we apply the network embedding method on the graph. DeepWalk [25] is a recently proposed method for learning the latent representations of nodes from truncated random walks in the network. DeepWalk combines random walk based proximity with the SkipGram model, a language model maximizing the co-occurrence probability among the words that appear within a window in a sentence. However, DeepWalk has certain weaknesses when being applied to our problem settings since the random walk based proximity it adopts does not consider the heterogeneity of a HIN. Inspired by the meta path-based proximity model in a HIN [26], in which a meta path is a sequence of node types with edge types in between modeling a particular relationship, we design three types of paths based on our spatial task-based network to capture the spatial information, task-related information and temporal information. Then a Path Constrained DeepWalk (PC-DeepWalk) algorithm is proposed to embed the network into a low-dimensional space, such that the original nodes of the network are represented as vectors. The three types of paths are designed as followed:

i) *Spatial path* is a path, denoted in the form of $C_1^{t_1} \xrightarrow{SPR} C_2^{t_2} \xrightarrow{SPR} ... \xrightarrow{SPR} C_i^{t_i}$ ..., which only consists of TC nodes and spatial proximity edges.

ii) *Task-related path* is a path in the form of $C_1^{t_1} \xrightarrow{TRR} B_1 \xrightarrow{TRR} C_2^{t_2} \xrightarrow{TRR} B_2 ... \xrightarrow{TRR} C_i^{t} \xrightarrow{TRR} B_i ...$, which contains TC nodes, ST nodes and task relevance edges.

iii) *Time-ordered task-related path* is a particular case of task-related path to capture the temporal trend of task releasing pattern, in which TC nodes are arranged by their time in an increasing order with the increasing rate of one time unit, i.e., $C_1^{t} \xrightarrow{TRR} B_1 \xrightarrow{TRR} C_2^{t+1} \xrightarrow{TRR} B_2 ... \xrightarrow{TRR} C_i^{t+i-1} \xrightarrow{TRR} B_i ...$

We then apply the λ-length random walk approach [25] along the proposed paths, which takes spatial task-based network G as input and a number of λ-length random walk sequences for each node as output. Getting the random walk sequences of each node, SkipGram model is leveraged to learn the representation vector of each node. Then we can estimate the task number in each cell through a regression algorithm by considering both historical data and other nodes. Formally,

$$N_i^{t_j+1} = \alpha N_i^{t_j} + (1-\alpha) \sum_{C_{i'}^{t_j} \in \mathbb{C}^{t_j}, i' \neq i} \frac{sim(C_i^{t_j}, C_{i'}^{t_j})}{\sum_{C_{i'}^{t_j} \in \mathbb{C}^{t_j}, i' \neq i} sim(C_i^{t_j}, C_{i'}^{t_j})} N_{i'}^{t_j} + \beta, \tag{2}$$

where $N_i^{t_j}$ is the task number of node $C_i^{t_j}$, $\mathbb{C}^{t_j}$ denotes all nodes in time t_j, and $sim(C_i^{t_j}, C_{i'}^{t_j})$ is the relevance between node $C_i^{t_j}$ and $C_{i'}^{t_j}$, which is computed by the dot product of their vectors. Therefore, $\sum_{C_{i'}^{t_j} \in \mathbb{C}^{t_j}, i' \neq i} \frac{sim(C_i^{t_j}, C_{i'}^{t_j})}{\sum_{C_{i'}^{t_j} \in \mathbb{C}^{t_j}, i' \neq i} sim(C_i^{t_j}, C_{i'}^{t_j})} N_{i'}^{t_j}$ represents the propagation effect between two nodes. α is a parameter controlling the contributions of historical data and other nodes. β is a parameter to avoid over-fitting. To obtain the optimal values of α and β, we leverage Stochastic Gradient Descent for training and use Mean Squared Error as the loss function.

B. Location Distribution Prediction

Once the task number $N_i^{t_j+1}$ in cell $C_i^{t_j+1}$ is obtained, we employ the KDE approach over locations of task samples in the cell to compute occurrence probability of potential tasks, where the task samples are composed of a set of historical tasks published in this cell, $\{s_1, s_2, ..., s_n\}$. The probability of potential task s being released in location $\mathbf{l} \in L_{C_i^{t_j+1}}$ in cell $C_i^{t_j+1}$ can be calculated as follows:

$$f(\mathbf{l}) = \frac{1}{|L_{C_i^{t_j+1}}| H^2} \sum_{s_i.l \in L_{C_i^{t_j+1}}} K(\frac{\hat{\mathbf{l}} - s_i.\hat{l}}{H}), \tag{3}$$

where $L_{C_i^{t_j+1}}$ is a set of locations of historical task samples in cell $C_i^{t_j+1}$, H is the bandwidth, each task location $s_i.l \in L_{C_i^{t_j+1}}$ (with latitude lat and longitude lon) is represented by $s_i.\hat{l} = (lat, lon)^T$, and function $K(\cdot)$ is a Gaussian kernel function given by $K(X) = \frac{1}{2\pi} exp(-\frac{1}{2} X^T X)$. The optimal bandwidth H can be computed in Equation 4.

$$H = \frac{1}{2} |L_{C_i^{t_j+1}}|^{-\frac{2}{3}} \sqrt{\hat{h}}^T \sqrt{\hat{h}}, \tag{4}$$

where $\hat{h}$ $(= \frac{1}{|L_{C_i^{t_j+1}}|} \sum_{s_i.l \in L_{C_i^{t_j+1}}} (s_i.\hat{l} - \bar{C}_i^{t_j+1})^2)$ is the variance of locations for all historical tasks in cell $C_i^{t_j+1}$ and $\bar{C}_i^{t_j+1}$ $(= \frac{1}{|L_{C_i^{t_j+1}}|} \sum_{s_i.l \in L_{C_i^{t_j+1}}} s_i.\hat{l})$ denotes the mean of these locations. According to the predicted task number $N_i^{t_j+1}$ in cell $C_i^{t_j+1}$, we set the top-$N_i^{t_j+1}$ locations with high probability as locations of the predicted tasks.

VI. Task Assignment

In this section, we will present the task assignment algorithms for solving the proposed PTA problem based on the current and predicted workers/tasks, with the aim of achieving the maximum task assignment. The basic idea is trying to find a union of one possible valid task sequence of all workers such that the number of assigned tasks can be maximized. In the sequel, we will first introduce the Maximal Valid Task Set (MaxVTS) (including both Location-specific and Route-specific MaxVTS) generation approach, in which the MaxVTS will be used throughout our algorithms. Then a greedy algorithm is proposed, which iteratively finds one "best" MaxVTS from the unassigned tasks for each worker until all the tasks are assigned or all the workers are exhausted. Finally, we design a graph partition based decomposition algorithm for task assignment, which can find the best union of MaxVTSs for all workers with optimal task assignment.

A. Maximal Valid Task Set Generation

1) Finding Reachable Tasks: Due to the constraint of workers' reachable distance and tasks' expiration time, each worker can only complete a small subset of tasks in the given time instance set, $T = \{t, t+1, ..., t+n\}$. Therefore, we firstly find the set of tasks that can be reached by each worker in the given time period T. The **location-specific reachable task subset** for a worker w, denoted as $L\text{-}RS_w$, should satisfy the following conditions: $\forall s \in L\text{-}RS_w$,

i) $c(w.l, s.l) \leq s.e - s.p$, and

ii) $c(w.l, s.l) \leq n + 1$, and

iii) $d(s_i.l, s_j.l) \leq w.range$, where $c(w.l, s.l)$ is the travel time from $w.l$ to $s.l$ and $d(a, b)$ is a given distance between location a and b.

As for **route-specified reachable task subset** (**$R\text{-}RS_w$**) for worker w, the following conditions should be satisfied: $\forall s \in R\text{-}RS_w$,

i) $c(w.l, s.l) \leq s.e - s.p$, and

ii) $c(w.l, s.l) \leq n + 1$, and

iii) $D(s_j.l, \mathbb{P}_w) \leq \frac{w.range}{2}$, where $D(a, b)$ is a given distance between location a and route b.

The above conditions guarantee that a worker can travel from her origin to the location of task s directly before it expires in the given time period T, where task s is located in her location/route-specific distance range.

2) Finding Maximal Valid Task Set: Given the reachable task set for each worker, we next find the set of MaxVTS. A dynamic programming algorithm is proposed to iteratively expand the sets of tasks in the ascending order of set size and find all MaxVTSs for a worker in each iteration. For each task in one set, we consider the scenario that it is finished in the end, and find all completed task sequences. Specifically, given a worker w, and a set of tasks $Q \subseteq RS_w$, we define opt(Q, s_j) as the maximum number of tasks completed by scheduling all the tasks in Q with constraints starting from $w.l$ and ending at $s_j.l$ within worker's reachable range. And R is denoted as the corresponding task sequence on Q to achieve this optimal value. We also use s_i to denote the second-to-last task before arriving at s_j in R, and R' to denote the corresponding task sequence for opt$(Q - \{s_j\}, s_i)$. Then opt(Q, s_j) can be calculated by Equation 5.

$$opt(Q, s_j) = \begin{cases} 1 & \text{if } |Q| = 1 \\ \max\limits_{s_i \in Q, s_i \neq s_j} opt(Q - \{s_j\}, s_i) + \delta_{ij} & \text{otherwise,} \end{cases} \tag{5}$$

$$\delta_{ij} = \begin{cases} 1 & \text{if } t(s_j.l) \leq s_j.e,\ t(s_j.l) \leq t + n, \\ 0 & \text{otherwise.} \end{cases}$$

$\delta_{ij} = 1$ means s_j can be finished after appending s_j to R' in the given time period $T = \{t, t+1, ..., t+n\}$. Based on Equation 5, we can obtain all the MaxVTSs for each worker.

When Q contains only one task s_i, the problem is trivial and opt$(\{s_i\}, s_i)$ is set to 1. When $|Q| > 1$, we need to search through Q to examine all possibilities of valid task sets and find the particular s_i that achieves the optimum value of $opt(Q, s_j)$. In Figure 1 and 2(b), taking w_1 with 4 reachable tasks $\{s_1, s_2, s_3, s_4\}$ as a case and computing the L-MaxVTSs for w_2, our algorithm starts by computing $opt(\{s_1\}, s_1) = 1$, $opt(\{s_2\}, s_2) = 1$, $opt(\{s_3\}, s_3) = 1$ and $opt(\{s_4\}, s_4) = 1$. For all the sets with size from 2 to 4, we iteratively compute the *opt* value and the corresponding R. For example, $opt(\{s_1, s_2\}, s_2) = 1$ since by following the task sequence (s_1, s_2), only s_1 can be finished, but $opt(\{s_1, s_2\}, s_1) = 2$ because by following (s_2, s_1), both s_1 and s_2 can be finished. The R-MaxVTSs for w_2 can be obtained in the same way.

B. Greedy Task Assignment

Once the MaxVTSs for each worker are obtained, a straightforward solution is to assign each worker with the maximal valid task set from the unassigned tasks, until all the tasks are assigned or all the workers are exhausted, which is called Greedy Task Assignment (GTA) algorithm since it does not consider the overall best strategy to assign tasks.

Algorithm 1: Greedy Task Assignment

```
  Input: $W, S$
  Output: A feasible assignment result $A$ and the corresponding
          number of assigned tasks $|A|$
1 $A \leftarrow \emptyset$;
2 for each worker $w \in W$ do
3     $Q_w \leftarrow \max\{MaxVTS(w, S)\}$;
4     $A = A \cup Q_w$;
5     $S = S - Q_w$;
6     $W = W - w$;
7 return $A$ and $|A|$;
```

We explain GTA in Algorithm 1. With a worker set W and task set S as input, it initializes an empty task assignment A (line 1). During each iteration, GTA begins to randomly select a worker $w \in W$ from the remaining ones to be assigned and finds the maximal MaxVTS from the unassigned tasks for the selected worker, in which the maximal MaxVTS is added into the current task assignment A (line 2-6). Finally, we find the maximal task assignment among all the iterations (line 7).

C. Optimal Task Assignment

The main computational challenge lies in huge search space when enumerating all possible combinations of the valid task sets of each worker, which increases exponentially with respect to the number of workers. However, in practice a worker shares the same tasks with only a few other workers who have similar or intersected travel routes. In the sequel, we first construct a worker dependency graph. By adopting a graph partition method on this graph and organizing the worker set of each subgraph in a tree structure, the problem is decomposed into multiple independent sub-problems. Then a depth-first search algorithm is devised to find the optimal task assignment.

1) Worker Dependency Graph Construction: A Worker Dependency Graph (WDG) is constructed based on the dependent/independent relations among workers, in which two workers are independent with each other if they share no reachable tasks and are dependent with each other otherwise. In particular, given a worker set W and task set S, a WDG, $G(V, E)$, is designed for encoding all the dependency relationship between workers, where each node $v \in V$ represents a worker $w_v \in W$, and each edge $e(u, v) \in E$ exists between u and v if the two workers w_u and w_v are dependent with each other.

2) Graph Partition: Subsequently, we apply a degree-k Graph Reduction-based (GR) method [27] to decompose workers' dependency relationship by partitioning the WDG graph. The result of graph partition contains a set of nodes, $X = \{X_1, ..., X_n\}$, which should satisfy the following conditions:

i) $\cup_{i \in n} X_i = V$, and

ii) $\forall (u, v) \in E, \exists X_i \in X$ containing both u and v, and

iii) if X_i, X_j and X_k are nodes, and X_k is on the path from X_i to X_j, then $X_i \cap X_j \subseteq X_k$.

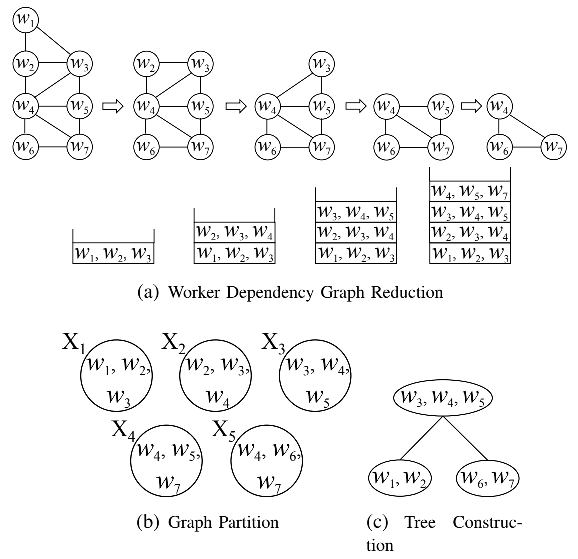
Specifically, degree-k GR is to reduce a graph into another simple graph with fewer vertices by removing the vertices whose degree is not more than k, in order to find the node set, X. The procedure is designed as follows:

i) Given a WDG and a specified degree i ($0 \leq i \leq k$), the vertex v with degree i is first identified, and we check whether all its neighbors form a clique. If not, we add the missing edges to construct a clique.

ii) The vertex v together with its neighbors, which is a clique, are pushed into a stack. This step is followed by removing v and the corresponding edges in the graph.

iii) All the cliques in the stack as well as the left clique that is not removed by the above reduction process will be the nodes (i.e., X) of the graph partition result.

We perform the vertex removing procedure above from deleting the vertices with degree 0 (i.e., isolated vertices) and then process it in the ascending order of vertex degree, until one of the following conditions is fulfilled: 1) the graph is reduced to a simple graph (e.g., a single triangle) or an empty set; 2) there exists no vertices with degree that is less than or equal to k. Figure 7(a) illustrates the reduction process for a given WDG, which starts with a degree-2 reduction by removing vertex w_1 and its edges. Vertex w_1 and its neighbors are then pushed in a stack. Subsequently, vertex w_2, w_3 and w_5 are removed respectively, following the same principle as of w_1. Finally, a single triangle is left, and the cliques of the graph can be found and output as the nodes of graph partition: $X = \{\{w_1, w_2, w_3\}, \{w_2, w_3, w_4\}, \{w_3, w_4, w_5\}, \{w_4, w_5, w_7\}, \{w_4, w_6, w_7\}\}$, as shown in Figure 7(b).



(a) Worker Dependency Graph Reduction

(b) Graph Partition

(c) Tree Construction

Fig. 7. Worker Partition

3) Tree Construction: According to the properties of graph partition, if two nodes do not share the same vertexes, the workers belonging to the two nodes are independent with each other. In this step, our goal is to organize the subsets of workers in a tree structure such that the sibling nodes are independent with each other, wherein we can solve the optimal assignment sub-problem on each sibling node independently. We next construct a balanced tree by the following Recursive Tree Construction (RTC) algorithm:

i) Remove the vertices in each node $X_i \in X$ from the WDG (i.e., G) in order to make G to be separated into a few components. We record the largest component as G_{max}.

ii) Pick the node X_{min} that leads to the least $|G_{max}|$ (the number of vertices in G_{max}) upon the completion of the previous loop. When there is a tie on $|G_{max}|$, the smallest X_i is picked as X_{min}. Then set X_{min} as the parent node for each output of the recursive procedure in step iii.

iii) Apply the degree-k GR algorithm on each sub graph by removing workers of X_{min} and recursively perform this algorithm on the output of degree-k GR algorithm.

iv) Return $N = X_{min}$ as the root node of this sub-tree.

With RTC algorithm, the final tree structure is depicted in Figure 7(c). After transforming the worker dependency graph into a tree structure, the depth-first search procedure can be applied to compute the suitable valid task set for each worker in the nodes of the tree in order to find the optimal assignment.

VII. Experiment

A. Experiment Setup

We conduct our experiments using two real datasets, Twitter-Foursquare (TF) dataset and gMission (GM) dataset, where Twitter-Foursquare dataset provides check-in data with category information and gMission [28] is a research-based general spatial crowdsourcing platform.

For TF dataset, the geo-tagged check-ins are used to simulate our problem, where the check-in dataset is collected from

TABLE II
EXPERIMENT PARAMETERS

Parameters	Values
Size of historical data $\|\mathcal{T}\|$ (i.e., percentage of the training location data)	20%, 40%, 60%, 80%, 100%
Number of tasks $\|S\|$ (TF)	1K, 2K, 3K, 4K, 5K
Number of tasks $\|S\|$ (GM)	300, 400, 500, 600, 700
Valid time of tasks $e - p$	1, 2, 3, 4, 5
Reachable distance of workers $range$	2km, 2.5km, 3km, 3.5km, 4km

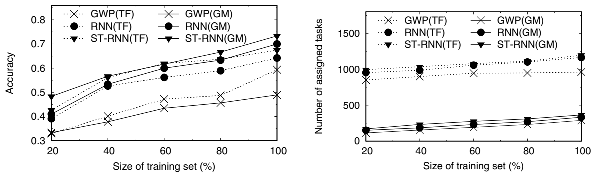
Twitter, for a period from September 2010 to January 2011. Since the original Twitter dataset does not contain the category information of venues, we extract the category information associated with each venue from Foursquare with the aid of its API. In total, the resulting dataset provides check-in data in the area of New York (with latitude from $40.231°$ to $41.231°$ and longitude from $-74.435°$ to $-73.435°$), which includes 29046 check-ins for 2056 users. When using this dataset in our experimental study, we assume the users are the workers in the 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. For each 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 checking in a spot is equivalent to accepting a task.

The GM dataset includes 532 workers and 713 tasks, wherein each worker has her location, arrival time and deadline, and each task is associated with a location, a release time, a deadline and a task description used to classify the tasks. Due to the lack of historical data for workers/tasks in gMission system, we generate workers/tasks that join the system in the history (i.e., a recent month) as follows. For each worker/task, we set her/its location as center and randomly produce her/its historical locations with Gaussian distribution, where her/its occurrence times are uniformly distributed in every day.

For both data sets, we simulate the trajectories of each worker in each day in the following way. The daily locations of a worker are fed into Simulation of Urban MObility (SUMO) to generate GPS trajectories, which are produced in a probabilistic manner. Moreover, we set the granularity of a time instance as one hour (i.e., 9:00am–10:00am), during which the task requests and available workers will be packed and input to our framework. We assign tasks to the suitable workers in 6 time instances (consisting of the current time instance and future 5 time instances) in the experiment. Table II shows our experimental settings, where the default values of all parameters are underlined. All the algorithms are implemented on an Intel Core i5-2400 CPU @ 3.10G HZ with 8 GB RAM.

B. Experiment Results

1) Performance of Worker Prediction: In this part, we evaluate the performance of worker prediction phase and its impact to the subsequent task assignment. We choose 70% location data of workers/tasks for training, 20% for testing and the remaining 10% as the validation set.



(a) Accuracy (b) Number of Assigned Tasks

Fig. 8. Performance of Workers' Location Prediction: Effect of $|\mathcal{T}|$

For location prediction of workers, two representative methods are compared with ST-RNN: 1) RNN [29]: estimating future locations with temporal dependency in workers' behavior sequence only; 2) GWP: Grid-based Worker Prediction [5]. To measure the performance of each model, we propose an accuracy rate, denoted by $acc(l) = \frac{\left|\tilde{l}|d(\tilde{l},l)\leq\varepsilon, l\in L^W_{\{t,\ldots,t+n\}}\right|}{\left|L^W_{\{t,\ldots,t+n\}}\right|}$, as the evaluation metric, where $\tilde{l}$ is the predicted location for true location l, $d(a,b)$ is Euclidean distance between point a and b, ε is a spatial deviation threshold (set to 0.5km), and $L^W_{\{t,\ldots,t+n\}}$ denotes all workers' locations occurred during $\{t,...,t+n\}$. We consider l is accurately predicted if it satisfies $d(\tilde{l},l) \leq \varepsilon$. Note that the time window widths of all the approaches are set to 4.

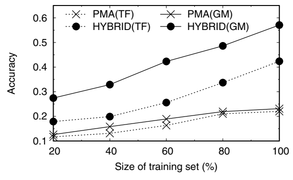
For effectiveness of task assignment based on the above prediction algorithms, we compare the number of actual assigned tasks, which are existing or correctly predicted, by applying the Optimal Task Assignment (OTA) algorithm. We conduct all the experiments on both TF and GM datasets.

Effect of $|\mathcal{T}|$. In the first set of experiment, we change the size $|\mathcal{T}|$ of training set and study their effect on workers' location prediction. From Figure 8(a), naturally the accuracy of all approaches increases when more training trajectories are used. Among these methods, ST-RNN achieves the highest accuracy rate followed by RNN and GWP in both TF and GM datasets. In Figure 8(b), the task assignment result heavily depends on the prediction accuracy since a better accuracy normally means more correctly predicted workers. ST-RNN performs best amongst all the methods for all values of $|\mathcal{T}|$, confirming the optimality of our proposed algorithm.

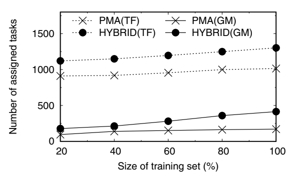
For route prediction evaluation, we introduce another accuracy rate, $acc(\mathbb{P})$, defined as the ratio between number of correctly predicted road links and the total road links. Then we compare our hybrid model (marked with HYBRID) with a baseline method, Pattern Matching Approach (PMA), by varying the size of training set.

As expected, the accuracies of both algorithms gradually increase as $|\mathcal{T}|$ grows (see Figure 9(a)). Our hybrid model makes significant improvement of accuracy over PMA, showing more benefits as $|\mathcal{T}|$ increases. On the task assignment aspect, Figure 9(b) demonstrates task assignment results are affected by prediction accuracy. PMA with low accuracy rate (e.g., $acc(\mathbb{P}) \leq 0.24$) assigns less tasks than HYBRID regardless of $|\mathcal{T}|$ in both datasets.

2) Performance of Task Prediction: In this set of experiments, we introduce two competitors, DeepWalk [25] and Grid-based Task Prediction [5] (GTP), and evaluate the lo-

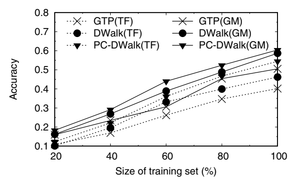


(a) Accuracy

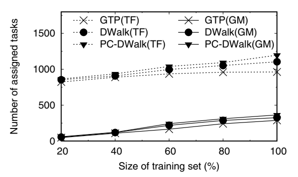


(b) Number of Assigned Tasks

Fig. 9. Performance of Workers' Route Prediction: Effect of $|\mathcal{T}|$

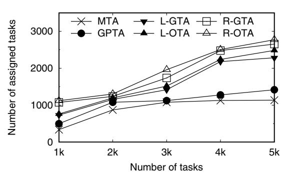


(a) Accuracy

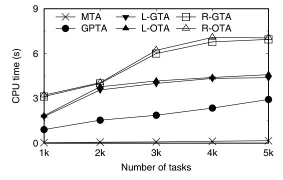


(b) Number of Assigned Tasks

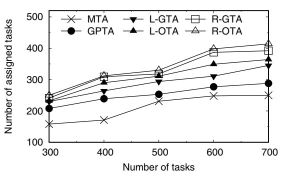
Fig. 10. Performance of Task Prediction: Effect of $|\mathcal{T}|$



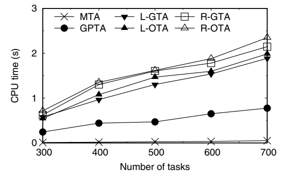
(a) Number of Assigned Tasks (TF)



(b) CPU Cost (TF)



(c) Number of Assigned Tasks (GM)



(d) CPU Cost (GM)

Fig. 11. Performance of Task Assignment: Effect of $|S|$

cation prediction accuracy of tasks using $acc(l)$. At the same time, we use the number of assigned tasks (generated by OTA) to measure the effectiveness of task assignment by varying $|\mathcal{T}|$.

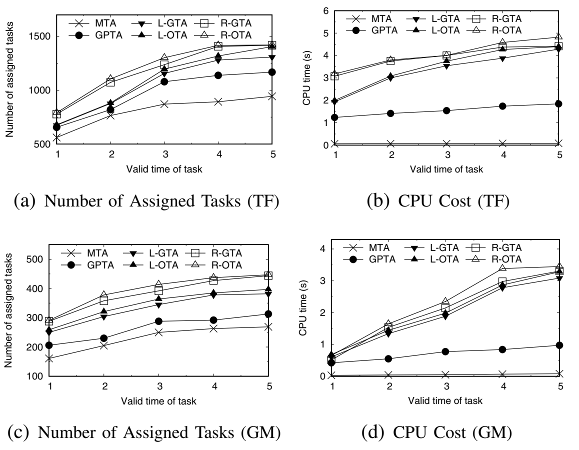
Effect of $|\mathcal{T}|$. In Figure 10(a), the accuracies of all prediction methods are improved with larger training data involved. We also observe that PC-DeepWalk can improve accuracy of location prediction compared with other two baseline approaches and generate the most accurate locations, which in turn leads to the largest number of assigned tasks as confirmed in Figure 10(b).

3) Performance of Task Assignment: In this part, we evaluate the effectiveness and efficiency of the task assignment approaches in terms of the overall number of assigned tasks and CPU time. Specially, the number of assigned tasks can measure the quality of task assignment strategies and the CPU time is given by the average time cost of performing task assignment at each time instance. We evaluate our proposed Greedy Task Assignment (GTA) and Optimal Task Assignment (OTA) algorithms based on workers' location/route prediction and tasks' prediction: Location-specific GTA (L-GTA), Location-specific OTA (L-OTA), Route-specific GTA (R-GTA) and Route-specific OTA (R-OTA). A straightforward approach, Maximum Task Assignment (MTA) [6] algorithm, which conducts the task assignment in current and future time instances separately without prediction, is introduced as a baseline algorithm. Moreover, we also implement a representative prediction-based task assignment algorithm, Grid-based Predictive Task Assignment (GPTA) with grid-based worker/task prediction [5], as our competitor. In GPTA, the quality score of assigning a worker to perform a task is set to 1 in order to get the maximal assigned tasks.

Effect of $|S|$. First, we investigate how the number of tasks affects the effectiveness and efficiency of task assignment. As expected, the numbers of assigned tasks for all algorithms gradually increase as $|S|$ grows in both TF and GM datasets, which is indicated in Figure 11(a) and 11(c). MTA generates the smallest assigned task set while R-OTA results the largest followed by R-GTA, L-OTA, L-GTA and GPTA. Not surprisingly, R-OTA and L-OTA generate more assigned tasks than their respective competitors (i.e., R-GTA and L-GTA) that use greedy task assignment strategy. Route-specific task assignment algorithms (i.e., R-OTA and R-GTA) assign more tasks than the location-specific methods (i.e., L-OTA and L-GTA) since workers have larger reachable range when performing tasks along the specified routes. In terms of running time, as shown in Figure 11(b) and 11(d), MTA is the fastest algorithm and almost not affected by $|S|$, while R-OTA is most time-consuming. R-OTA (L-OTA) runs slower than R-GTA (L-GTA) mainly because of the extra time cost for building the tree to be searched. Although GPTA is more efficient than our proposed approaches, it assigns less tasks.

Effect of $e-p$. Next we study the effect of the valid time of tasks, $e-p$. As illustrated in Figure 12(a) and 12(c), naturally the numbers of assigned tasks generated from all approaches increase when the valid time of tasks become longer. This is due to the fact that a worker has more chance to be assigned the tasks with more relaxed valid time. Similar to the previous results, our proposed route-specific task assignment methods can achieve more assigned tasks than location-specific task assignment methods, and both of them outperform GPTA and MTA, which confirms the superiority of our proposed algorithms. We can see from Figure 12(b) and 12(d), the running times of all methods increase for longer valid times of tasks, since there are more worker-and-task assignments to process.

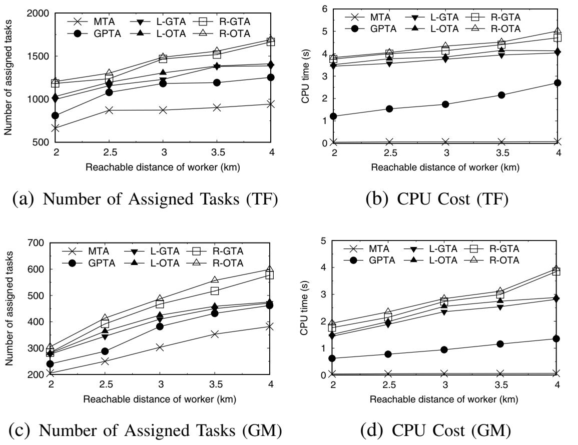
Effect of $range$. As depicted in Figure 13(a) and 13(c), the numbers of assigned tasks generated by all approaches have a growing tendency as $range$ being enlarged, with the similar reason of the effects of tasks' valid time, i.e., the larger the workers' reachable regions are, the more chance the SC server has to assign the workers more tasks. In addition, L/R-OTA and L/R-GTA outperform the others for all values of $range$, which demonstrates the effectiveness of our proposed algorithms again. The CPU cost of all the approaches increases with the enlarged $range$ (see Figure 13(b) and 13(d)), since the number of available tasks to be assigned in a time instance grows when $range$ gets larger, which in turn leads to longer



(a) Number of Assigned Tasks (TF) (b) CPU Cost (TF)

(c) Number of Assigned Tasks (GM) (d) CPU Cost (GM)

Fig. 12. Performance of Task Assignment: Effect of $e - p$



(a) Number of Assigned Tasks (TF) (b) CPU Cost (TF)

(c) Number of Assigned Tasks (GM) (d) CPU Cost (GM)

Fig. 13. Performance of Task Assignment: Effect of $range$

time cost.

VIII. Conclusion

In this paper we propose a novel data-driven framework, called Data-driven Predictive Spatial Task Assignment (DP-STA), to assign the tasks to workers by considering both current and future workers/tasks that enter the spatial crowdsourcing system dynamically. We propose different strategies to predict the spatio-temporal distribution of future workers and tasks. Then, we design a greedy algorithm to efficiently assign tasks and a graph partition based decomposition algorithm to find the global optimal task assignment. Extensive empirical study based on real datasets confirms our proposed framework can significantly improve the effectiveness of task assignment.

Acknowledgment

This work is partially supported by Natural Science Foundation of China (No. 61972069, No. 61836007, No. 61832017, and No. 61532018).

References

[1] 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*, vol. 9, no. 12, pp. 1053–1064, 2016.

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

[3] 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*, 2017, pp. 1009–1020.

[4] M. Lv, Q. Wang, and Z. Yuan, "Personal trajectory pattern matching for future route prediction," *Journal of Computational Information Systems*, vol. 10, no. 1, pp. 197–204, 2014.

[5] P. Cheng, X. Lian, L. Chen, and C. Shahabi, "Prediction-based task assignment in spatial crowdsourcing," in *ICDE*, 2017, pp. 997–1008.

[6] L. Kazemi and C. Shahabi, "Geocrowd: Enabling query answering with spatial crowdsourcing," in *SIGSPATIAL*, 2012, pp. 189–198.

[7] H. To, L. Fan, T. Luan, and C. Shahabi, "Real-time task assignment in hyperlocal spatial crowdsourcing under budget constraints," in *PerCom*, 2016, pp. 1–8.

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

[9] Y. Zhao, J. Xia, G. Liu, H. Su, D. Lian, S. Shang, and K. Zheng, "Preference-aware task assignment in spatial crowdsourcing," in *AAAI*, 2019.

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

[11] Y. Cui, L. Deng, Y. Zhao, B. Yao, V. W. Zheng, and K. Zheng, "Hidden poi ranking with spatial crowdsourcing," in *KDD*, 2019.

[12] 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.

[13] Y. Tong, L. Chen, Z. Zhou, H. V. Jagadish, L. Shou, and W. Lv, "Slade: A smart large-scale task decomposer in crowdsourcing," *TKDE*, vol. PP, no. 99, pp. 1588–1601, 2018.

[14] P. Cheng, X. Lian, L. Chen, and J. Han, "Task assignment on multi-skill oriented spatial crowdsourcing," *TKDE*, vol. 28, no. 8, pp. 2201–2215, 2015.

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

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

[17] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," in *SIGMOD*, 2000, pp. 1–12.

[18] Q. Liu, S. Wu, L. Wang, and T. Tan, "Predicting the next location: A recurrent model with spatial and temporal contexts," in *AAAI*, 2016, pp. 194–200.

[19] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," in *UAI*, 2009, pp. 452–461.

[20] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 399–421, 1988.

[21] T. Wang, D. Zhang, X. Zhou, X. Qi, H. Ni, H. Wang, and G. Zhou, "Mining personal frequent routes via road corner detection," *Transactions on Systems Man & Cybernetics Systems*, vol. 46, no. 4, pp. 445–458, 2016.

[22] J. Pei, J. Han, B. Mortazaviasl, H. Pinto, Q. Chen, U. Dayal, and M. C. Hsu, "Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth," in *ICDE*, 2001, pp. 215–224.

[23] M. Morzy, "Mining frequent trajectories of moving objects for location prediction," in *MLDM*, 2007, pp. 667–680.

[24] E. C. Delmelle and J. C. Thill, "Urban bicyclists: Spatial analysis of adult and youth traffic hazard intensity," *Transportation Research Record Journal of the Transportation Research Board*, vol. 2074, no. 2074, pp. 31–39, 2008.

[25] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *KDD*, 2014, pp. 701–710.

[26] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu, "Pathsim: Meta path-based top-k similarity search in heterogeneous information networks," *VLDB*, vol. 4, no. 11, pp. 992–1003, 2011.

[27] F. Wei, "Tedi: Efficient shortest path query answering on graphs," in *SIGMOD*, 2010, pp. 99–110.

[28] Z. Chen, R. Fu, Z. Zhao, Z. Liu, L. Xia, L. Chen, P. Cheng, C. C. Cao, Y. Tong, and C. J. Zhang, "gmission: A general spatial crowdsourcing platform," *VLDB*, vol. 7, no. 13, pp. 1629–1632, 2014.

[29] Y. Zhang, H. Dai, C. Xu, J. Feng, T. Wang, J. Bian, B. Wang, and T. Y. Liu, "Sequential click prediction for sponsored search with recurrent neural networks," in *AAAI*, 2014, pp. 1369–1375.