Effective Task Assignment in Mobility Prediction-Aware Spatial Crowdsourcing (Technical Report)

Huiling Li†, Yafei Li†, Wei Chen†, Shuo He†, Mingliang Xu†, Jianliang Xu‡
†*School of Computer and Artificial Intelligence, Zhengzhou University, Zhengzhou, China*
‡*Department of Computer Science, Hong Kong Baptist University, Hong Kong, China*
{ieyfli, heshuo, iexumingliang}@zzu.edu.cn, {huilingli, chenweizzu}@gs.zzu.edu.cn, xujl@comp.hkbu.edu.hk

***Abstract*—With the proliferation of mobile devices, spatial crowdsourcing has emerged as a promising paradigm for facilitating location-based services, encompassing various applications across academia and industry. Recently, pioneering works have attempted to infer workers' mobility patterns from historical data to improve the quality of task assignment. However, these studies have overlooked or under-examined issues such as the dynamic mobility patterns of crowd workers, especially in the context of newcomers, the misalignment between the objectives of mobility prediction and task assignment, and the effective utilization of predicted mobility patterns. In this paper, we investigate a problem we term Task Assignment in Mobility Prediction-aware Spatial Crowdsourcing (TAMP). To address the TAMP problem, we first propose a task-adaptive meta-learning algorithm, which trains a set of specific meta-knowledge for workers' mobility prediction models through game theory-based learning task clustering and meta-training within each cluster. Then, we design a task assignment-oriented loss function and develop a task assignment algorithm that incorporates prediction performance, prioritizing assignments with higher confidence of completion. Extensive experiments on real-world datasets validate that our proposed methods can effectively improve the quality of task assignment.**

***Index Terms*—spatial crowdsourcing, task assignment, mobility prediction, meta learning**

I. Instruction

With the development and widespread use of mobile devices, spatial crowdsourcing (SC) has become increasingly popular, serving as a novel solution for location-based services in both academia (e.g., gMission [1], TVDP [2], and FloraVision [3]) and industry (e.g., OpenStreetMap [4], TaskRabbit [5], and Waze [6]). In a typical SC platform, users dynamically post spatial tasks to the platform based on their needs, and the platform is responsible for assigning tasks to suitable crowd workers who can selectively accept the assigned tasks.

Generally, crowd workers prefer to perform spatial tasks in conjunction with their daily routines to earn additional income. However, their detailed daily itineraries are not always known to the platform in advance due to the inherent uncertainty of their day-to-day activities [7]. Consequently, the ability to infer and predict these concealed daily patterns from their historical mobility data is essential for devising effective task assignments. Despite pioneering efforts [8]–[10] to predict workers' mobility using historical data, these approaches do not address the cold start problem that arises from new workers continually joining the platform. Instead, they often resort to a random strategy for dealing with new workers or select only active workers for their experimental settings [8]. Incorporating new workers is a critical aspect of spatial crowdsourcing. Overlooking this factor can negatively affect worker retention rates and reduce their participation in crowdsourcing initiatives, potentially threatening the platform's ongoing success and long-term sustainability. Furthermore, existing studies [8], [10], [11] only consider the unidirectional impact of mobility prediction on task assignment, while ignoring the effect of task assignment on mobility prediction. Such inadequate consideration of how task assignment and mobility prediction work together can hinder the effective use of predicted mobility patterns.

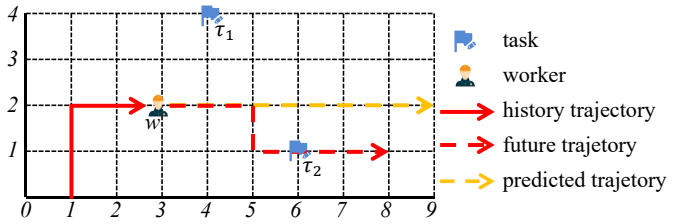


Fig. 1. A running example of TAMP. There are one crowd worker w and two spatial tasks τ_1 and τ_2 at the current time. The solid red line represents the historical trajectory of w, the red dotted line indicates the true future trajectory, and the brown dotted line shows the predicted trajectory of w based on a mobility prediction model using the historical trajectory. Without mobility prediction, we can only assign tasks based on w's current location (e.g., in a distance first strategy). In this case, τ_1 will be assigned to w. While based on the predicted trajectory, τ_2 will be assigned to w, as τ_2 is closer to the predicted trajectory. Obviously, τ_2 is more suitable for w according to w's real trajectory (ground truth). As such, although there is still a gap between predicted trajectory and future trajectory, it does indicate the future movement pattern of worker, which is beneficial for task assignment.

In this paper, we explore a novel problem, which we refer to as *Task Assignment in Mobility Prediction-aware Spatial Crowdsourcing (TAMP)*. A running example of TAMP is shown in Fig. 1. More specifically, TAMP involves the platform assigning tasks based on predicted worker mobility patterns, while workers retain the choice to accept tasks according to their actual movements. The primary objective of TAMP is threefold: to maximize the total number of tasks completed, to minimize the rejection rate of tasks by workers, and to reduce the overall cost of task completion. To achieve these goals, TAMP addresses the following three main challenges, tied to predicting workers' mobility patterns (*Challenge I*), empowering mobility prediction from the view of task assignment (*Challenge II*), and fully utilizing predicted mobility patterns in task assignment (*Challenge III*), respectively.

Challenge I pertains to the dynamic mobility patterns among workers, which arise from three main factors. Firstly, the variability in travel preferences among different workers directly contributes to the dynamism in their mobility patterns. Secondly, workers' opportunistic behavior leads to unpredictable engagement with platforms, adding to the dynamic nature of the observed mobility. Thirdly, the constant influx of new workers introduces novel and previously unexplored mobility patterns to the platform. To tackle this challenge, we introduce worker-specific mobility prediction, where learning a single worker's mobility pattern is treated as an individual learning task unit. Specifically, we develop a task-adaptive meta-learning algorithm. This algorithm clusters learning tasks using distribution, spatial, and gradient-based learning paths as representative features. It also integrates the individual objectives of the learning tasks (i.e., the workers) into the multi-level clustering process via a game-theoretic approach. Subsequently, Model-Agnostic Meta-Learning (MAML) [12] is applied within each cluster.

Challenge II addresses the disconnect between the objectives of mobility prediction models and task assignment. In existing literature on prediction-aware spatial crowdsourcing [8], [13]–[15], prediction models are typically trained using only the ground truth of raw data for orientation. However, from a task assignment standpoint, this approach is not always ideal. The main issue is that the impact of the discrepancy between a worker's predicted and actual trajectory at a specific point, especially in terms of the worker's ability to complete a particular task, varies depending on the location of the spatial task. Moreover, when traditional loss functions are used, they treat all discrepancies equally in terms of their influence on updating the parameters of the prediction model. To address this, we propose a task assignment-oriented loss function that adjusts the Mean Squared Error (MSE) loss based on the distribution of historical spatial tasks.

Challenge III involves the effective utilization of uncertain predicted mobility. Since perfectly accurate predictions of workers' mobility are unattainable, the critical issue is the effective use of these imperfect predictions in the context of task assignment within prediction-aware spatial crowdsourcing. A performance evaluation metric that assesses mobility prediction models from the task assignment perspective is urgently needed. To address this, we introduce an evaluation metric called the *matching rate*, which transforms the performance of mobility prediction models into the probability of workers completing tasks. Building on this, we propose a task assignment algorithm that incorporates prediction performance.

To summarize, the main contributions of this paper are as follows:

- We introduce a novel problem named TAMP, where the SC platform dynamically predicts workers' mobility and assigns them appropriate tasks.
- We develop a worker-specific mobility prediction model that incorporates task-adaptive meta-learning through game-theory-based learning task clustering. Additionally, we design a task assignment-oriented loss function to better align mobility prediction with task assignment.
- We propose a performance-involved task assignment algorithm that effectively integrates the prediction model's performance into the task assignment process, giving priority to assignments with a higher likelihood of successful completion.
- We validate the effectiveness of our proposed methods through experiments on real-world datasets.

The remainder of this paper is organized as follows. Section II formally defines the TAMP problem. Section III details our proposed solutions. We evaluate the performance of our proposed algorithms in Section IV. Section V reviews the related works. Finally, we conclude this paper in Section VI.

II. Problem Statement

In this section, we first present the system model of this work. Then, we formally define the problem of Task Assignment in Mobility Prediction-aware Spatial Crowdsourcing. Table I summarizes the notations used in this paper.

TABLE I
Summary of Main Notations

Notation	Definition
τ, T	a spatial task, a set of spatial tasks
w, W	a crowd worker, a set of crowd workers
$r, \hat{r}$	a worker's predicted daily routine
M	a task assignment plan
Γ_i	the learning task corresponding to worker w_i
$\mathbb{V}_i$	the POI sequence of worker w_i
$\mathbb{G}_i$	the k-step gradient path of learning task Γ_i
Sim_s	the spatial feature similarity
Sim_l	the learning path similarity
Sim_d	the distribution similarity
d	the acceptable detour of worker

In a spatial crowdsourcing platform, there are many part-time workers who are willing to provide appropriate services during their daily leisure time [8], [9], [16]. Inspired by this, we study mobility prediction-aware spatial crowdsourcing in this work. Figure 2 shows its overall framework, consisting of three parties: spatial crowdsourcing platform, task requesters, and crowd workers. Generally, task requesters publish spatial tasks to the SC platform (step 1), after which the SC platform performs batch-based task assignment [17]–[19] according to the predicted mobilities of workers, i.e., assigns suitable workers for the published tasks (step 2). Then, crowd workers report on the execution of tasks, whether they accept or reject the assigned tasks (step 3). Finally, the SC platform reports the status of tasks to the requesters (step 4). In addition, the SC platform's operation mainly consists of two stages, i.e., offline training and online task assignment. Generally, during the offline training stage, it employs historical data and trains the mobility prediction model for workers. For the online task assignment stage, it first predicts the workers' mobility leveraging the trained model and then performs task assignment in batch-based mode. Thereafter, we formally elaborate on the system model and give the definitions involved.

Definition 1 (Spatial Task)**.** *A spatial task is denoted as a tuple $\tau = (l, t)$, where $\tau.l$, and $\tau.t$ represent the target location and deadline of τ, respectively.*

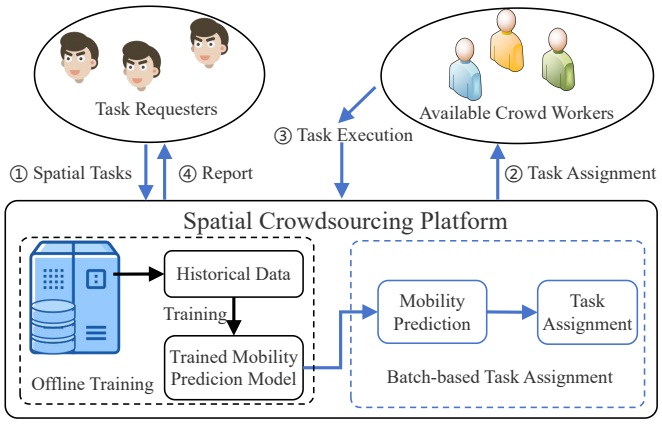


Fig. 2. Framework for mobility prediction-aware spatial crowdsourcing.

In the SC platform, the spatial tasks randomly arrive at the platform and each spatial task requires a worker to reach the target location $\tau.l$ before its deadline $\tau.t$ so that this spatial task can be completed. Also, a spatial task can only be assigned to one worker at a time.

Definition 2 (Crowd Worker)**.** *A crowd worker is denoted as a tuple $w = (r, l, d)$, where $w.r$ and $w.l$ is the historical routine and current location of w, respectively. $w.d$ represents the maximum detour distance that w can accept to complete a spatial task. A routine $r = \{(l_1, t_1), (l_2, t_2), \cdots, (l, t)\}$ is a series of locations with timestamps.*

In general, a crowd worker w comes to the platform dynamically and moves according to his/her daily routine, which is unknown to the platform in advance. In the mobility prediction-aware spatial crowdsourcing, the platform predicts w's possible routine $w.\hat{r}$ according to the historical routines of w and assigns tasks to suitable workers accordingly. While in terms of worker w, he/she can decide whether to accept the assigned task according to his/her actual itinerary and acceptable detour distance $w.d$. Workers do not upload their daily route plans. Instead, when they are online, they merely share their current location with the platform. Meanwhile, we assume that w will only accept tasks that result in a detour no greater than $w.d$ upon completion to ensure their daily routine is completed on time. This assumption is reasonable since crowd part-time workers are always willing to accept tasks closely along with their daily routines so that they do not necessitate a substantial alteration of their initial itineraries.

Definition 3 (Mobility Prediction)**.** *Given worker w's historical routine $w.r$, mobility prediction aims to infer his/her next mobility routine $w.\hat{r}$.*

Given a training set $\mathbb{D} = \{(r_1, y_1), (r_2, y_2), \cdots, (r_m, y_m)\}$ of historical trajectories from worker w, where r_i $(1 \leq i \leq m)$ is a sub trajectory with fixed length seq_{in} sampled from the historical route $w.r$ of worker w, y_i is the sub trajectory immediately following r_i with fixed length seq_{out}, and m is the number of sampled trajectory pairs. The mobility prediction trains a personalized prediction model on dataset $\mathbb{D}$ to learn the mobility patterns of worker w so that it can continuously forecast w's subsequent mobility routine $w.\hat{r}$.

Definition 4 (Task Assignment)**.** *Given a set of spatial tasks T and a set of crowd workers W, a task assignment M between T and W can be defined as a set of pairs with form (τ, w), where $\tau \in T$, $w \in W$. More specifically, the assignment M is associated with a set of assignment pairs M' accepted by workers, and each tuple $(\tau, w) \in M'$ is associated with a real cost d_c represents the detour that w takes to complete τ.*

Note that a task assignment M is available only when $\forall\ \tau \in T\ \&\ \forall\ w \in W$, both $(\tau, \cdot)$ and $(\cdot, w)$ only appears once in M. Based on the above system model, we can formally define the TAMP problem.

Definition 5 (TAMP Problem)**.** *Given the unassigned spatial task set T and the crowd worker set W with their predicted routines, the TAMP problem aims to find a task assignment M that can maximize $|M'|$, while minimizing the rejection rate $\frac{|M|-|M'|}{|M|}$ and workers' average cost $\frac{\sum_{(\tau,w)\in M'} d_c}{|M'|}$.*

III. Solution

In this section, we present the solution for the TAMP problem, including worker-specific mobility prediction, task assignment-oriented loss function, and prediction performance-involved task assignment.

A. Overview

Solving the TAMP problem necessitates the development of a model that can accurately predict the mobility of workers, including new arrivals. To this end, we introduce worker-specific mobility prediction, which leverages task-adaptive meta-learning to train a set of personalized initialization parameters for each worker's mobility prediction model. Consequently, the model can swiftly converge to the desired performance level, which is especially effective for the mobility prediction of new arrival workers. Second, we devise a weighted function, grounded in historical spatial task data, to re-weight the computation of losses with the aim of a more targeted optimization process for task assignment. Third, we concretize the influence of the mobility prediction model's performance on task assignment and design the prediction performance-involved task assignment algorithm.

B. Worker Specific Mobility Prediction

In general, worker-specific mobility prediction is addressed by task-adaptive meta-learning, which can be regarded as a decomposition of the learning task set [20] and trains a set of initialization parameters for each learning task. Here, a learning task, denoted by Γ to distinguish from the spatial task, represents predicting a worker's daily mobility based on his/her historical trajectory data. Each learning task Γ_i is uniquely associated with a worker w_i. Subsequently, we delve into the details of the proposed algorithm.

Game Theory-Based Task Adaptive Meta-Learning. The typical MAML [12] assumes that all the tasks come from the same distribution. Since workers' mobility patterns are

generally variable, this assumption is not always satisfied. To overcome the disadvantage of typical MAML, we propose a novel meta-learning framework, i.e., a game theory-based task-adaptive meta-learning approach. Overall, we first cluster all the learning tasks according to the spatial features, learning paths, and distribution similarity by modeling the clustering problem as a multi-player strategy game. Then, standard MAML is adopted among each task cluster, training a common initialization parameter for learning tasks in this cluster. The cluster in which a learning task is located directly affects the performance of its corresponding worker's mobility model. However, traditional clustering algorithms only consider the overall goal and ignore each element's goal (i.e., the worker's individual goal) involved in clustering. This oversight results in an unequal treatment of all elements. As such, we model clustering as an n-player strategy game with the intention of finding Nash equilibrium. Since, in the Nash equilibrium, all workers reach their maximum achievable goal in their current situation, this approach fully considers the individual goals of the workers corresponding to the learning tasks. Meanwhile, the factors used to represent learning tasks during clustering are also of great significance, and we introduce these three factors in what follows:

- *Spatial feature.* We use the Point-of-Interests (POIs) to model the spatial features of learning tasks. Given a learning task Γ_i, the POI sequence corresponding to the collection of historical data in the spatial task performed by worker w_i can be expressed as $\mathbb{V}^{(i)} = \{v_1, v_2, \ldots, v_n\}$, where $v_i = \langle x_i, y_i, a_i \rangle$, x_i, y_i and a_i represent the latitude, longitude, and type of POI v_i. $\mathbb{V}^{(i)}$ is regarded as the spatial feature of learning task Γ_i.
- *Learning path.* We adopt the k-step gradient during the parameter update process of each learning task to represent the learning path of the task. Given a learning task Γ_i, its k-step gradient can be represented as $\mathbb{Z}^{(i)} = \{z_1, z_2, \ldots, z_k\}$, where each z_j in $\mathbb{Z}^{(i)}$ is a vector representing the gradient at j-th step when training a meta-learner on Γ_i.
- *Distribution.* The data distribution of a learning task Γ is a direct representation of it.

In responding to the above three factors, we calculate the similarity among learning tasks.

Spatial feature similarity. Given two learning tasks Γ_i and Γ_j related to w_i and w_j, based on the kernel estimation method [21], [22], we calculate the spatial feature similarity as:

$$Sim_s(\Gamma_i, \Gamma_j) = Norm(\frac{1}{|\mathbb{V}^{(i)}||\mathbb{V}^{(j)}|} \sum_{a=1}^{|\mathbb{V}^{(i)}|} \sum_{b=1}^{|\mathbb{V}^{(j)}|} K_h(v_a, v_b)), \tag{1}$$

where $K_h(v_a, v_b)$ is the kernel function used in [22] and $Norm(\cdot)$ is a function that normalizes the the feature similarity to $[0, 1]$ with 0 representing completely dissimilar, 1 representing completely similar.

Learning path similarity. Given the learning path $\mathbb{Z}^{(i)}$ and $\mathbb{Z}^{(j)}$ of two learning tasks Γ_i and Γ_j, we take the average cosine similarity to represent the similarity between them, which can be calculated as:

$$Sim_l(\Gamma_a, \Gamma_b) = \frac{1}{k} \sum_{i=1}^{k} cos(z_i^{(a)}, z_i^{(b)}). \tag{2}$$

Distribution similarity. Following an existing work [23], we adopt Wasserstein distance to evaluate the similarity between two distributions. Given two distributions, the distribution similarity can be calculated as:

$$Sim_d(\Gamma_a, \Gamma_b) = \frac{1}{\inf\limits_{q \sim \Pi(\Gamma_a, \Gamma_b)} \mathbb{E}_{(x,y)\sim q}[\|x - y\|]}. \tag{3}$$

As such, the quality of a learning task cluster G can be calculated as follows:

$$Q(G) = \begin{cases} \frac{\sum_{\Gamma_i \in G} \sum_{\Gamma_j \in G \setminus \{\Gamma_i\}} Sim_d(\Gamma_i, \Gamma_j)}{|G|(|G|-1)}, & |G| > 1 \\ \gamma, & |G| = 1 \\ 0, & |G| = 0 \end{cases}, \tag{4}$$

where $Sim_d(\Gamma_i, \Gamma_j)$ can also be replaced by $Sim_s(\Gamma_i, \Gamma_j)$ or $Sim_l(\Gamma_i, \Gamma_j)$, depending on which indicator is selected when calculating the quality of the learning task cluster. $\gamma \in (0, 1)$ is a hyperparameter representing the utility of clustering when only a learning task is in the cluster, which will be further explained in the subsequent content of this work. Intuitively, the higher the quality of the cluster, the higher the similarity between the learning tasks and the more similar the mobility patterns between the workers within the cluster.

Based on the above similarity calculation function between any two learning tasks, we model the problem of clustering the learning tasks as an n-player strategy game, which can be donated as a three-entry tuple $\mathcal{P} = (N, \{ST_i\}_{i \in N}, \{U_i\}_{i \in N})$. Where N is the set of players (each player corresponds to a learning task, and we also use Γ_i to represent the player), ST_i is the strategy set of player i. Each strategy indicates that a player chooses to join a different learning task cluster. Then, $\vec{st} = (st_1, st_2, \ldots, st_n)$ is a joint strategy, where $st_i \in ST_i$ is a specific strategy of player i. Furthermore, the utility function of player i can be calculated as:

$$U_i(\vec{st}) = U_i(st_i, \vec{st}_{-i}) = u(\Gamma_i, G_j) = Q(G_j) - Q(G_j \setminus \{\Gamma_i\}), \tag{5}$$

where G_j is the learning task group that G_i would like to join by choosing strategy st_i. Intuitively, $U_i(\vec{st})$ donates the change in $Q(G_j)$ after learning task Γ_i joins task cluster G_j. As such, the hyperparameter γ in Equ. 4 actually means that only clusters with a quality greater than γ will be retained.

In addition, since the goal of the strategy game $\mathcal{P}$ is Nash equilibrium, we first prove the existence of Nash equilibrium in game $\mathcal{P}$ and propose Theorem 1.

Theorem 1. *The n-player strategy game $\mathcal{P}$ has Nash equilibrium, which can be found through the best-response framework.*

Proof. Please refer to Appendix A-A for details. □

Meanwhile, as can be seen from the proof process, the validity of Theorem 1 is independent of the choice of similarity calculation function in Equ. 4. Regardless of which factor is chosen in the clustering process, Theorem 1 holds.

By modeling the clustering problem as an exact potential game, clustering becomes finding the Nash equilibrium, where each player achieves their individual goals for that exact potential game. This means that game-based clustering considers the goals of the individuals (learning tasks, workers) participating in the clustering compared with typical clustering algorithms like k-means.

Given that multiple factors can be used to calculate the similarity between learning tasks, we propose a multi-level clustering algorithm. Prior to this, we first defined a learning task tree for storing the results of multi-level clustering.

Definition 6 (Learning Task Tree)**.** *A learning task tree is a multi-forked tree, where the node in this tree can be represented as a tuple $T^t = (G, \mathcal{CH}, fr, \theta)$. Here G is the learning task cluster of this node, $\mathcal{CH}$ is the list of all the children node of T^t, fr is T^t's father node, and θ indicates the initialization weight of the mobility prediction model corresponding to this node.*

Algorithm 1: Game Theory-based Multi-level Learning Task Clustering (GTMC)

Input: A learning task set N, k, an ordered list of similarity functions F^s and the list of corresponding threshold values Θ
Output: the learning task tree T_0^t

1 $T_0^t.G \leftarrow N$;
2 $NodeQueue \leftarrow \{(T_0^t, 0)\}$;
3 **while** $NodeQueue \neq \emptyset$ **do**
4 $(T^t, j) \leftarrow$ queue header from $NodeQueue$;
5 $\mathcal{G}_{sub} \leftarrow$ initialize clusters by k-medoids algorithm [24] on $T^t.G$ with $1/F_j^s$ as the distance function between any two learning tasks;
6 **while** *not in a Nash equilibrium* **do**
7 **foreach** *player* $i \in T^t.G$ **do**
8 $st_i \leftarrow$ the best-response strategy for Γ_i with the highest utility $U_i(st_i, \vec{st}_{-i})$;
9 **if** st_i *is not* $st_i^{'}$ **then**
10 remove Γ_i from task cluster $G_{sub}^{'}$;
11 add Γ_i to task cluster G_{sub};
/* G_{sub} and $G_{sub}^{'}$ are the clusters that i chooses to join in strategies st_i and $st_i^{'}$ */
12 remove all G_{sub} without learning tasks from $\mathcal{G}_{sub}$;
13 **if** $|\mathcal{G}_{sub}| > 1$ **then**
14 **foreach** $G_{sub} \in \mathcal{G}_{sub}$ **do**
15 $T_{new}^t \leftarrow (G_{sub}, \emptyset, T^t, T^t.\theta)$;
16 add T_{new}^t to $T^t.\mathcal{CH}$;
17 **if** $j+1 \leq |F^s|$ *and* $Q(G_{sub}) < \Theta_j$ **then**
18 add $(T_{new}^t, j+1)$ to the end of $NodeQueue$;
19 **return** T_0^t;

Algorithm 1 illustrates the pseudo of Game Theory-based Multi-level Learning Task Clustering (GTMC). It takes a learning task set N and an ordered list F^s of similarity functions as input with the corresponding threshold list Θ and outputs the constructed learning task tree T_0^t. Specifically, it initializes the root node of the learning task tree and a queue, the element of which is in the form of (T^t, j), where j is used to indicate the similarity function employed for further clustering the learning tasks in $T^t.G$ (lines 1-2). Then, with the help of $NodeQueue$, we employ the best-response framework (lines 5-11) to cluster learning tasks and build a learning task tree (lines 3-18). For each cluster that needed to be clustered further (line 4), it first initializes k clusters by calling k-medoids [24] algorithm (line 5). Then, we employ the best-response framework to iteratively select the strategy with the highest utility for each player and update the corresponding clusters until a Nash equilibrium is found (lines 6-11). For each sub-cluster G_{sub} with no learning tasks in it, we remove it from $\mathcal{G}_{sub}$ (line 12). After removing the sub-clusters without learning tasks, if the number of sub-clusters in $\mathcal{G}_{sub}$ is larger than 1, we build new learning task nodes based on each sub-cluster and add them as the child nodes of T^t (lines 13-16). If $j+1 \leq |F^s|$ and the quality of cluster $Q(G_{sub}$ is less than the threshold Θ_j, which indicates that this sub-cluster needs to be further clustered to the next level, we add $(T_{new}^i, j+1)$ to queue $NodeQueue$ (lines 17-18). Finally, if $NodeQueue$ is empty, clustering ends, and the algorithm returns the final learning task tree T_0^t (line 19).

Complexity Analysis. Please refer to Appendix B-A for details.

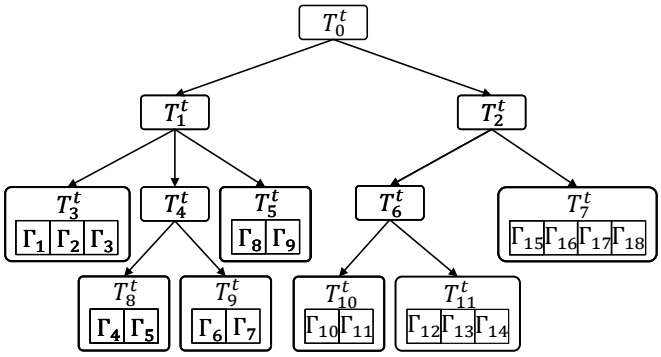


Fig. 3. An example of the learning task tree.

Fig. 3 shows an example of a learning task tree. Only leaf nodes carry training data in a learning task tree, while non-leaf nodes only store model initialization parameters. Then, we introduce how to train the initialization parameters for the learning tasks corresponding to each task cluster based on the learning task tree and propose Task Adaptive Meta-Learning (TAML) in this work. As shown in Algorithm 2, it takes a learning task tree T^t, the meta-learning rate α, and the adapt rate β as input. If T^t doesn't have any child node, it calls Algorithm 3 and returns the result of Algorithm 3 (lines 1-2). Otherwise, it traverses all of T^t's child nodes, recursively calls itself, and accumulates the returned loss $\mathcal{L}_{T^t}(f_\theta)$ (lines 3-4). Finally, it updates $T^t.\theta$ based on the average gradient of all child nodes of T^t and return loss $\mathcal{L}_{T^t}^{avg}(f_\theta)$ (lines 5-7).

Complexity Analysis. Please refer to Appendix B-B for details.

In addition, Algorithm 3 illustrates the specific training process based on MAML for a given learning task cluster. It takes a learning task tree T^t, meta-learning rate α, and adapt

Algorithm 2: Task Adaptive Meta-learning (TAML)

Input: A learning task tree T^t, meta-learning rate α, and adapt rate β
Output: Average loss $\mathcal{L}_{T^t}^{avg}(f_\theta)$

1 **if** $T^t.\mathcal{CH} = \emptyset$ **then**
2 | **return** Meta-Training(T^t, α, β);
3 **foreach** $T_i^t \in T^t.\mathcal{CH}$ **do**
4 | $\mathcal{L}_{T^t}(f_\theta) \leftarrow \mathcal{L}_{T^t}(f_\theta) +$ TAML(T_i^t, α, β);
5 $\mathcal{L}_{T^t}^{avg}(f_\theta) \leftarrow \frac{1}{|T^t.\mathcal{CH}|}\mathcal{L}_{T^t}(f_\theta)$;
6 $T^t.\theta \leftarrow T^t.\theta - \alpha\nabla\mathcal{L}_{T^t}^{avg}(f_\theta)$;
7 **return** $\mathcal{L}_{T^t}^{avg}(f_\theta)$;

rate β as input. Generally, it iteratively samples learning tasks, performs adaptive weight updates, calculates query losses, and performs meta updates (lines 1-9). In each iteration, the algorithm first samples a batch of m learning task G_{sam} from $T^t.G$ (line 2). For each sampled learning task $\Gamma_i \in G_{sam}$, it adapts k steps on the model f_{θ_i} initialized by $T^t.\theta$ on the support set of learning task Γ_i (lines 4-7). Afterward, on the query set of Γ_i, the algorithm calculates the query loss of model f_{θ_i} that has undergone k step adaptation (line 8). After the adaptive update for each batch is completed, it calculates the gradient based on the average query loss of m sampled learning tasks and updates the model initialization parameter $T^t.\theta$ accordingly (line 9). When all iterations are completed, it calculates the average query loss $\mathcal{L}_{T^t}^{avg}(f_\theta)$ for multiple iterations and returns $\mathcal{L}_{T^t}^{avg}(f_\theta)$ (lines 10-11).

Complexity analysis. Please refer to Appendix B-C for details.

Subsequently, we elaborate on leveraging the trained learning task tree for the mobility prediction of newly arrived workers. Given the learning task T^t corresponding to a newly arrived worker w, we proceed with a depth-first postorder traversal of the learning task tree, wherein we calculate the average similarity between T^t and the learning tasks encompassed within each node. Then, we initialize the mobility prediction model of w with the parameters from the most similar node and conduct model training based on this initialization.

Discussion. Our meta-learning algorithm is model-agnostic and compatible with any machine learning or deep learning model updatable via gradient descent. Instead of designing a specific worker mobility model, we focus on creating a meta-learning approach that optimizes initialization parameters after a few rounds of adaptive training based on which mobility prediction can achieve desirable performance. Nonetheless, a predictive model for worker mobility is indispensable; hence, we implement an encoder-decoder model [25] based on Long Short-Term Memory (LSTM) neural networks [26]: LSTM-Encoder-Decoder model.

C. Task Assignment Oriented Loss Function

The general objects of the mobility prediction models are directly minimizing the mean square error (MSE) [27], [28], cross-entropy [13], [29] or KL-divergence [30] between the real trajectory and the predicted one. However, we observe that the above objects do not always align with the goal of task assignment. For example, the role of each point on a worker's trajectory in task assignment is different (influenced by spatial task distribution), but it is indeed the same when calculating losses. As such, when mobility prediction is oriented toward task assignment, developing a novel loss function becomes imperative. In this section, to address the gap between the objectives of the prediction model and task assignment, we propose a task assignment-oriented loss function for the mobility prediction model.

Algorithm 3: Meta-Training

Input: A learning task cluster T^t, meta-learning rate α, and adapt rate β
Output: Average query loss $\mathcal{L}_{T^t}^{avg}(f_\theta)$

1 **while** *not done* **do**
2 | Sample a batch of m learning tasks G_{sam} from $T^t.G$;
3 | **foreach** $\Gamma_i \in G_{sam}$ **do**
| | /* Adapt k steps on task Γ_i */
4 | | $\theta_i \leftarrow T^t.\theta$;
5 | | **for** $i \leftarrow 1$ ***to*** k **do**
6 | | | Compute adapt loss $\mathcal{L}_{\Gamma_i}^s(f_{\theta_i})$;
7 | | | $\theta_i \leftarrow \theta_i - \beta\nabla\mathcal{L}_{\Gamma_i}^s(f_{\theta_i})$;
8 | | Compute query loss $\mathcal{L}_{\Gamma_i}^q(f_{\theta_i})$;
9 | $T^t.\theta \leftarrow T^t.\theta - \alpha\nabla\frac{1}{|m|}\sum_{i=1}^{m}\mathcal{L}_{\Gamma_i}^q(f_{\theta_i})$;
10 Compute the average query loss $\mathcal{L}_{T^t}^{avg}(f_\theta)$ for multiple iterations;
11 **return** $\mathcal{L}_{T^t}^{avg}(f_\theta)$;

From the perspective of task assignment, the predicted trajectory should exhibit an influence that is as identical or similar as feasible to the influence of the actual trajectory on task assignment. The primary factor influencing spatial task assignment is the distance from the spatial task to the locations on the worker's trajectory. For example, if the detours for a worker w to complete a certain task τ calculated based on his/her predicted trajectory and real trajectory are the same, then from the perspective of assigning worker w to perform task τ, the predicted trajectory and the real trajectory are equivalent, even though there are differences between them. Based on the above observation, when calculating the loss, we expect to pay more attention to the trajectories surrounding which tasks are prone to occur. As such, we design a weighted loss function based on MSE. Specifically, given the predicted trajectory $\vec{r}$, ground-truth r and the weighted function $f_w(\cdot,\cdot)$, the loss function can be represented as:

$$\mathcal{L}_T = \frac{1}{|r|}\sum_{i=1}^{|r|} f_w(l_i)\cdot(l_i - \hat{l}_i)^2. \tag{6}$$

And the weighted function $f_w(\cdot,\cdot)$ is designed as:

$$f_w(l_i) = \kappa\cdot\frac{|\{\tau|dis(\tau,l_i) < d^q, \tau\in\mathcal{T}\}|}{\rho^t} + \delta, \tag{7}$$

where $\mathcal{T}$ represents the collection of all the historical tasks, d^q is a hyperparameter indicating the range of tasks that affect the weight calculation at location l_i, ρ^t is the number of tasks in a unit space (the unit space is the area of a circular space with radius d^q), $\kappa \in (0,1)$ and the offset $\delta \in \mathbb{R}_+$ are used to control the impact of historical data on loss calculation.

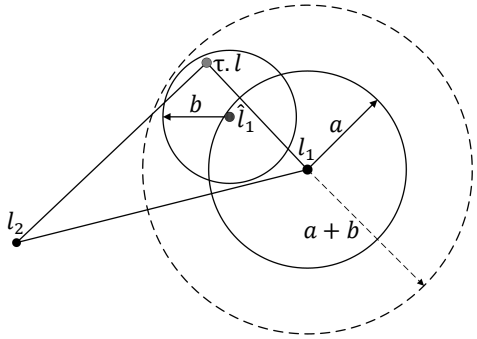


Fig. 4. An example illustrating Lemma 1.

Here, only the spatial relationship between the workers' trajectories and tasks is considered through the hyperparameter d^q, while the temporal relationship is overlooked. The rationality lies in the fact that the matching between available workers and tasks taken into account is typically within a specific time window [16], [31], [32]. Consequently, the temporal correlation is confined within a relatively narrow range, rendering our main focus on the spatial relationship.

D. Task Assignment with Workers Predicted Mobility

Next, we design the task assignment algorithm for the TAMP problem. In contrast to conventional task assignment problems, assigning tasks to workers with predicted mobility patterns necessitates considering the impact of the mobility prediction model's performance on the algorithm. However, general metrics like MSE and RMSE for regression analysis in machine learning are not always suitable for designing task assignment algorithms. As such, an evaluation metric that assesses the performance of mobility prediction models from the perspective of task assignment is in demand. Then, following [13], we adopt the matching rate to evaluate the performance of the mobility prediction model for each worker. Concurrently, the matching rate computation methodology from [13] is tailored to accommodate the TAMP problem explored in this work.

Definition 7 (Matching Rate)**.** *Given the real routine r and the predicted routine $\hat{r}$ of the same worker, the match rate $MR(r,\hat{r}) = \frac{1}{|r|}\sum_{i=1}^{|r|} match(l_i,\hat{l}_i)$ represents the average number of matched locations in the real routine and predicted routine, where $match(l_i,\hat{l}_i) = \begin{cases} 1, & dis(l_i,\hat{l}_i) \le a \\ 0, & dis(l_i,\hat{l}_i) > a \end{cases}$, where $a \in \mathbb{R}_+$ is the hyperparameter representing the distance threshold for matching between the real location and the predicted location.*

Evidently, in the case where a is determined, a higher value of the matching rate indicates superior performance of the mobility prediction model, whereas a lower value signifies inferior performance. Moreover, we regard $MR(r,\hat{r})$ as the probability that the predicted location $\hat{l}_i$ of l_i in routine r falls within a circle centered at l_i with a radius of a. Consequently, drawing upon the illustrative example presented in Fig 4, we formulate Lemma 1.

Lemma 1. *On the condition that the probability of $dis(l_1,\hat{l}_1) \le a$ is $MR(r,\hat{r})$, if a task τ happens to fall within the area centered at $\hat{l}_1$ with b as radius and $a+b \le d/2$, the expected probability that worker w completes task τ without violating worker's detour constraint is also $MR(r,\hat{r})$, when w is at location l_1. Here, d is the acceptable detour distance of worker w.*

Proof. Please refer to Appendix A-B for details. □

Regarding the deadline constraint for task τ, we propose Lemma 2, which is similar to Lemma 1.

Lemma 2. *On the condition of Lemma 1, if $a+b < d^t = sp*(\tau.t - t_c)$, the expected probability that worker w completes task τ without violating the deadline constraint of task τ is also $MR(r,\hat{r})$. Here sp is the speed of worker w and t_c represents the current timestamp.*

Proof. Please refer to Appendix A-C for details. □

To integrate the detour constraint of workers and the deadline constraint of tasks, we summarize Lemma 1 and 2 and propose Theorem 2.

Theorem 2. *On the condition of Lemma 1 and Lemma 2, if $a+b < min(d/2, d^t)$, the expected probability that worker w completes task τ without violating constraints is $MR(r,\hat{r})$.*

Proof. Please refer to Appendix A-D for details. □

In general, through Theorem 2, we transform the matching rate of the predicted and real locations in the worker's routine into the probability of a worker completing a specific task without violating any constraints.

Then, we design the Prediction Performance-Involved Task Assignment Algorithm (abbreviated PPI) with its pseudo-code presented in Algorithm 4. The main idea is that based on Theorem 2, priority is given to tasks more likely to be completed. Specifically, it takes a task set T, a worker set W with every worker's mobility prediction performance, and a parameter ϵ as input and outputs a task assignment plan M. In the first stage, It initializes M_c and $\mathcal{B}$ as empty sets and traverses all combinations of workers and tasks (lines 1-11). During the traversal process, it calculates the distance between each location in worker w's predicted trajectory and $\tau.l$ adding the distance $dis(\hat{i},\tau.l)$ that satisfies the condition in Theorem 2 to B (lines 4-7). Next, for cases where $|B|*w.MR \ge 1$, we add $(\tau, w, 1/minB)$ to M_c, otherwise we add (B,τ,w) to $\mathcal{B}$, where $minB$ represents the minimal distance in B (lines 8-11). Then, it calls the KM algorithm [33], [34] on M_c to obtain the result of maximum-weight matching (line 12). Next, the task assignment of the second phase begins (lines 13-27), during which an auxiliary variable $counter$ is first initialized to 0, and M_c is set to $\emptyset$ (line 13). It traverses all $(B,\tau,w) \in \mathcal{B}$ in the reverse order of the value of $|B|*w.MR$ (lines 14-24). During traversal, if B is not $\emptyset$, it adds $(\tau, w, 1/minB)$ to M_c and counts with variable $counter$ (lines 15-17). Otherwise, it breaks the traversal loop (lines 18-19). If $counter$ reaches the threshold ϵ, it calls the KM algorithm on M_c (lines 20-21). Then it traverses the (τ, w) pair in M_f and removes elements

Algorithm 4: Prediction Performance-Involved Task Assignment

Input: A task set T, a worker set W with every worker's mobility prediction performance $w.MR$, a parameter $\epsilon \in \mathbb{N}_+$
Output: A task assignment plan M

1 $M_c \leftarrow \emptyset, \mathcal{B} \leftarrow \emptyset$;
2 **foreach** $\tau \in T$ **do**
3 **foreach** $w \in W$ **do**
4 $B \leftarrow \emptyset$;
5 **foreach** $\hat{l}_i \in w.\hat{r}$ **do**
6 **if** $dis(\hat{l}_i, \tau.l) + a < min(d/2, d^t)$ **then**
7 $B \leftarrow B \cup \{dis(\hat{i}, \tau.l)\}$;
8 **if** $|B| * w.MR \geq 1$ **then**
9 $M_c \leftarrow M_c \cup \{(\tau, w, 1/minB)\}$;
10 **else**
11 $\mathcal{B} \leftarrow \mathcal{B} \cup \{(B, \tau, w)\}$;
12 $M \leftarrow$ call KM algorithm [33], [34] on M_c;
13 $counter \leftarrow 0$, $M_c \leftarrow \emptyset$;
14 **foreach** $(B, \tau, w) \in \mathcal{B}$ *in the reverse order of the value of* $|B| * w.MR$ **do**
15 **if** $B \neq \emptyset$ **then**
16 $M_c \leftarrow M_c \cup \{(\tau, w, 1/minB)\}$;
17 $counter \leftarrow counter + 1$;
18 **else**
19 break;
20 **if** $counter = \epsilon$ **then**
21 $M_f \leftarrow$ call KM algorithm on M_c;
22 **foreach** $(\tau, w) \in M_f$ **do**
23 remove all elements related to τ and w from $\mathcal{B}$;
24 $M \leftarrow M \cup M_f, counter \leftarrow 0, M_c \leftarrow \emptyset$;
25 **if** $M_c \neq \emptyset$ **then**
26 $M_f \leftarrow$ call KM algorithm on M_c;
27 $M \leftarrow M \cup M_f$;
28 **foreach** *unassigned task* $\tau \in T$ **do**
29 **foreach** *unassigned worker* $w \in W$ **do**
30 $dis^{min} = \min_{\hat{l} \in w.\hat{r}} dis(\tau.l, \hat{l})$;
31 **if** $dis^{min} \leq min(d/2, d^t)$ **then**
32 $M_c \leftarrow M_c \cup \{(\tau, w, dis^{min})\}$;
33 $M_f \leftarrow$ call KM algorithm on M_c;
34 $M \leftarrow M \cup M_f$;
35 **return** M;

related to τ or w from $\mathcal{B}$, after which the matching results M_f is merged to M, $counter$ is reset to 0, and M_c is reset as $\emptyset$ (lines 22-24). Next, if there are still elements in M_c (whether due to the loop breaking at line 19 or the $counter$ not reaching ϵ at the end of the loop), we call the KM algorithm and merge the matching results M_f to M (lines 25-27).

The first and second stages focus on matching (τ, w) pairs with expected completion probabilities that can be determined using the performance of the mobility prediction model according to Theorem 2. However, for those (τ, w) pairs without expected completion probabilities, there is still a possibility of completion. As such, in the third stage (lines 28-34), for those unassigned tasks and workers, we add the (τ, w) pairs that satisfy the detour constraint and deadline constraint (i.e. $dis^{min} \leq min(d/2, d^t)$ and since the platform cannot calculate the worker's real detour, $dis^{min} \leq d/2$ is used to represent that the detour constraint is satisfied) calculated based solely on the predicted trajectory to M_c (lines 28-32). Then, the KM algorithm is called on M_c for matching (lines 33-34). Finally, after finishing all three assignment stages, the algorithm returns the final assignment plan M (line 35).

Discussion. In algorithm 4, we divide task assignment into multiple stages based on the probability of a specific worker completing a particular task supported by Theorem 2. At the end of each stage, the KM algorithm is called to solve the maximum weight matching. Intuitively, in scenarios where the trajectories of workers are fully known, this approach may result in a quality decrease of the task assignment plan, as the overall matching is decomposed into the union of results on multiple sub bipartite graphs. However, when task assignment is performed based on workers' imprecise mobility trajectories, since PPI gives priority to the assignment of tasks with a higher likelihood of successful completion, it contributes to reducing the task rejection rate, enhancing completion rates, and minimizing detours.

IV. Experiments

In this section, we evaluate the performance of our proposed mobility prediction algorithm and task assignment algorithm. All the approaches are implemented in Python3, and experiments were run on an Intel i9-9900K CPU @3.6 GHz with NVIDIA GeForce RTX 2070 GPU and 32 GB RAM.

A. Experiment Settings

Datasets. We evaluate our proposed approaches on two groups of real datasets, where Porto[1] (Gowalla [35]) and Didi [36] (Foursquare [37]) datasets are used to simulate crowd workers and spatial tasks, respectively. For more details about datasets and preprocessing, please refer to Appendix C-A.

Evaluation Metrics. First, we evaluate the performance of mobility prediction models using four metrics: (1) Root Mean Square Error (RMSE), (2) Mean Absolute Error (MAE), (3) Matching Rate (MR) and (4) Training Time (TT). RMSE and MAE are commonly used measurement indicators. MR is the matching rate defined in Def. 7. TT is the running time during the model training. Second, to evaluate the task assignment-oriented loss function and quality of task assignment, we use four metrics to measure effectiveness and efficiency. Specifically, we measure the effectiveness with (1) completion ratio, the ratio between the number of completed and total tasks; (2) average cost, the average detour that workers take to complete tasks; and (3) rejection ratio, the ratio between the number of rejected and total assignments, reflecting the performance of task assignment plans in meeting the real constraints of workers and indirectly reflecting the effectiveness of the mobility prediction model. We measure the efficiency of the task assignment algorithm with running time.

Parameter Settings. Table II describes the main parameter settings varied in the experiments. The valid time of tasks

[1]https://www.kaggle.com/c/pkdd-15-predict-taxi-service-trajectory-i/overview

is randomly sampled from a time interval. For example, the interval [3, 4] indicates that the valid time of tasks is between 3 and 4 time units, with each time unit representing 10 minutes. The time window for dividing task assignment batches is set to 2 minutes, and γ is set to 0.2. Regarding other hyper-parameters involved in algorithms, we strive to tune them to be optimal.

TABLE II
PARAMETER SETTINGS

Parameter	Value
seq_{out}	**1**, 2, 3
seq_{in}	1, **5**, 10
worker's detour d	2km, 4km, **6km**, 8km, 10km
# of spatial tasks	1K, 2K, **3K**, 4K, 5K
valid time of tasks	[1, 2], [2, 3], **[3, 4]**, [4, 5], [5, 6]

Compared Algorithms. (1) In terms of mobility prediction, we compare four algorithms, including our GTTAML and its variant.

- MAML [12]: is a widely used optimization-based meta-learning algorithm that does not cluster tasks but performs meta-training on all learning tasks.
- CTML [38]: is a clustered task-aware meta-learning algorithm, which clusters learning tasks by soft K-means according to features of input data and learning paths represented by parameter update trajectories.
- GTTAML: uses our GTMC to cluster learning tasks and performs MAML-based meta training in each cluster.
- GTTAML-GT: is the variant of GTTAML, in which only the k-means algorithm is used during multi-level clustering of learning tasks.

Since the task assignment-oriented loss function is designed to favor task assignment, the loss function used in evaluating the performance of the mobility prediction models is the MSE loss.

(2) Regarding task assignment, we compare the proposed algorithm and its variants with both self-designed comparison algorithms and algorithm proposed in other work.

- Upper Bound (UB): we implemented an algorithm that checks the constraints based on the worker's real trajectory, constructs a bipartite graph with the reciprocal of the real detour as the weight, and calls the KM algorithm to solve the maximum weight matching. The result obtained by this algorithm can be regarded as the upper bound of mobility prediction-aware task assignment, and the rejection rate of workers is bound to be 0.
- Lower Bound (LB): we also implemented an algorithm that generates the bipartite graph only based on workers' current location and performs task assignment by KM algorithm, the results of which are regarded as the lower bound of mobility prediction-aware task assignment.
- PPI: is the algorithm 4 and task assignment-oriented loss function is used in training mobility prediction models.
- KM: generates the bipartite graph according to the third stage of algorithm 4 and calls the KM algorithm [33], [34] to obtain the task assignment plan.
- GGPSO: is a genetic algorithm proposed in [8], which can optimise the current solution through crossover, mutation, and selection. It is also designed for task assignment in mobility prediction scenarios.
- KM-loss and PPI-loss: in contrast to KM and PPI, the loss function utilized in mobility prediction is the MSE loss, not our task assignment-oriented loss function.

B. Experimental Results

In this section, we analyze the experimental results in depth to validate the effectiveness of our approaches from two aspects: mobility prediction and task assignment.

(1) Performance of Mobility Prediction. To assess the performance of our mobility prediction algorithm, an ablation study was first conducted regarding the clustering algorithm and the factors considered when clustering learning tasks. Then, we varied the input sequence length seq_{in} and output sequence length seq_{out} in mobility prediction.

TABLE III
EFFECT OF LEARNING TASK CLUSTERING ON DATASET PORTO

cluster algorithm	cluster factor			metric			
	Sim_d	Sim_s	Sim_l	RMSE	MAE	MR	TT
GTMC	✓	✗	✗	0.9176	0.7939	0.4242	2262.6
	✗	✓	✗	0.9264	0.8020	0.4087	2412.7
	✗	✗	✓	0.9593	0.8428	0.3640	2228.4
	✓	✓	✗	0.9052	0.7856	0.4305	3285.5
	✓	✓	✓	**0.8937**	**0.7711**	**0.4446**	3987.1
k-means	✓	✗	✗	0.9663	0.8611	0.3846	1420.6
	✗	✓	✗	0.9736	0.8555	0.4011	1538.4
	✗	✗	✓	1.0880	0.9613	0.3253	2097.3
	✓	✓	✗	0.9643	0.8555	0.3927	1732.7
	✓	✓	✓	**0.9428**	**0.8369**	**0.4020**	2277.2

Ablation Study on Clustering Algorithms and Factors. Table III summarizes all experimental results when the clustering algorithm and factors differ. Specifically, when only a single clustering factor is considered, regardless of whether the clustering algorithm is GTMC or k-means, the effectiveness of the distribution feature is the best, followed by the spatial feature and learning path feature. The reason is that the distribution feature of data is the most direct representation of learning tasks, so it is the most effective one. Since this work focuses on predicting workers' mobility patterns in cities, the spatial feature also performs well. Furthermore, as we continue to sequentially add clustering factors on top of the distribution feature, the model's performance remains on an upward trend. This indicates that the three features are all effective. It also leads to increased training time, because taking more clustering factors into account results in more clusters within our hierarchical clustering framework. Furthermore, more clusters result in longer training time according to the time complexity of algorithms 2 and 3. Since the meta training is offline, such training time is tolerable, allowing for improved mobility prediction performance. Therefore, in subsequent experiments, these three clustering factors are all taken into account, and the order of similarity functions corresponding to these three factors in the similarity function list F^s of the GTMC algorithm is Sim_d, Sim_s, and Sim_l. Regarding different clustering algorithms, GTMC is always better than using only k-means for multi-level clustering. This

validates that modeling the clustering as a strategy game and considering the individual goals of workers during the clustering process confers advantages in mobility prediction.

TABLE IV
EFFECT OF seq_{in} AND seq_{out} ON DATASET PORTO

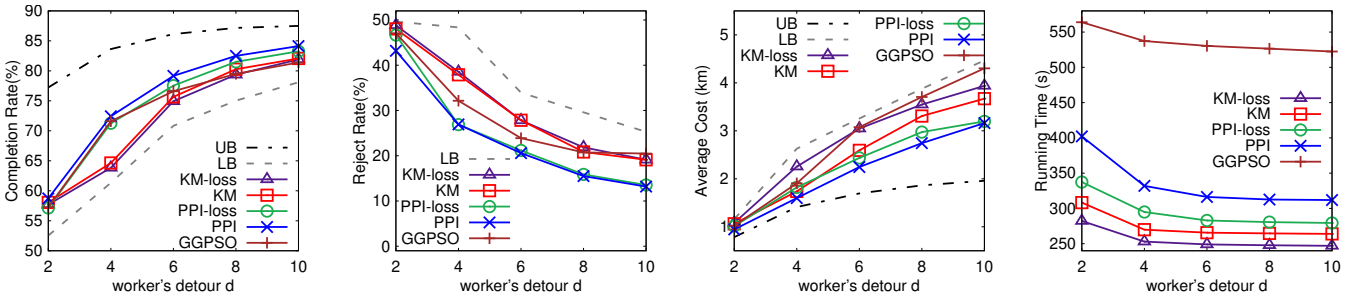
seq_{in}	metric	MAML	CTML	GTTAML-GT	GTTAML
1	RMSE	1.0332	0.9664	0.9317	**0.9063**
	MAE	0.9210	0.8590	0.8044	**0.7793**
	MR	0.2997	0.3600	0.4234	**0.4396**
	TT	1361.7	1643.8	1424.1	2531.1
5	RMSE	0.9722	0.9437	0.9428	**0.8937**
	MAE	0.8697	0.8215	0.8369	**0.7711**
	MR	0.3621	0.3881	0.4020	**0.4446**
	TT	2091.4	2577.1	2277.2	3987.1
10	RMSE	0.9466	0.9216	0.8991	**0.8976**
	MAE	0.8436	0.7967	0.7805	**0.7723**
	MR	0.3858	0.4309	0.4325	**0.4338**
	TT	2517.1	3718.0	3255.4	5624.3
seq_{out}	metric	MAML	CTML	GTTAML-GT	GTTAML
1	RMSE	0.9722	0.9437	0.9428	**0.8937**
	MAE	0.8697	0.8215	0.8369	**0.7711**
	MR	0.3621	0.3881	0.4020	**0.4446**
	TT	2091.4	2577.1	2277.2	3987.1
2	RMSE	1.1911	1.0517	1.1390	**1.0100**
	MAE	1.0422	0.9054	0.9557	**0.8691**
	MR	0.0936	0.3209	0.3358	**0.3431**
	TT	2090.6	2870.8	2620.4	4660.3
3	RMSE	1.1900	1.2082	1.1943	**1.1664**
	MAE	1.0162	1.0316	0.9818	**0.9646**
	MR	0.2129	0.2552	0.2815	**0.3051**
	TT	2404.3	3427.0	2984.3	5273.8

Effect of seq_{in} and seq_{out}. Table IV presents the performance of four algorithms, MAML, CTML, GTTAML-GT, and GTTAML, when varying seq_{in} and seq_{out}. Overall, regardless of how seq_{in} and seq_{out} change, our GTTAML displays the best performance. The reasons are as follows. Compared to MAML, its advantage lies in training more personalised model initialization parameters for each worker by clustering learning tasks. By incorporating more practical features into the clustering process, its performance surpasses CTML. The experimental results also illustrate that the gradient-based learning path representation is more effective than the parameter-based one on the dataset used in this work. Additionally, considering the individual goals of workers in the clustering is the primary reason for its better performance than GTTAML-GT. In addition, as seq_{in} increases, the performance of four algorithms exhibits a trend of improvement because the enlargement of seq_{in} enables the model to reference more input data during prediction. However, when seq_{in} increases from 5 to 10, the performance of the GTTAML slightly declines. This could be attributed to workers' mobility patterns being more strongly correlated with their most recent moving trajectories. Moreover, as we varied seq_{out} from 1 to 3, the performance of all four algorithms shows a decreasing trend. The reason is that predicting the mobility patterns of workers for longer periods is a more difficult task. This is acceptable in practice because the detour workers can tolerate is limited, making long-term predictions of workers' mobility patterns less significant. With the increase of seq_{in} and seq_{out}, the algorithms' training time increase. MAML has the shortest training time, followed by CTML, GTTAML, and GTTAML-GT in most cases. Also, such training time is tolerable because the meta training is offline.

(2) Performance of Task Assignment. By varying the worker's detour d, the number of spatial tasks and the valid time of tasks, we evaluated the performance of our task assignment-oriented loss function and proposed task assignment algorithm.

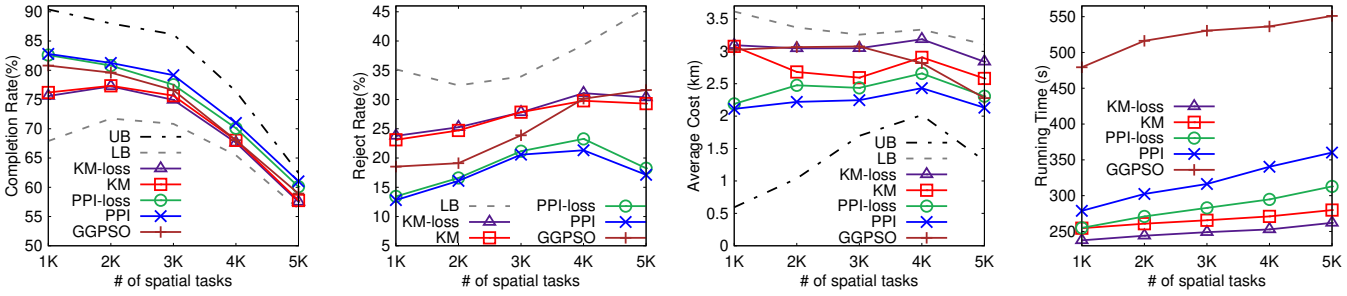
Effect of worker's detour d. As illustrated in Fig. 5, we compare five algorithms, PPI, PPI-loss, KM, KM-loss, and GGPSO, with the upper bound (UB) and lower bound (LB). Overall, the PPI algorithm performs best in the three effectiveness metrics: completion rate, rejection rate, and average cost. Specifically, with the increase in the acceptable detour d of workers, the task completion rates of all algorithms show an increasing trend. This is because the increase in d provides workers with more opportunities to accept tasks, which is also why the rejection rate shows a decreasing trend. Meanwhile, as tasks that require longer detours become more likely to be accepted, the average cost for workers to complete tasks exhibits an increasing trend. By comparing KM with KM-loss as well as PPI with PPI-loss, the effectiveness of our task assignment-oriented loss function can be verified, which improves the task completion rate, reduces the task rejection rate, and especially reduces the average cost. The reason is that our self-designed loss function incorporates task distribution into the calculation of loss, enabling the mobility prediction model to update its parameters in a more conducive direction to task assignment. Compared to KM, PPI shows significantly better performance in terms of quality metrics, particularly in rejection rate. This validates the effectiveness of our strategy, which prioritizes assigning tasks with higher certainty of completion by workers. Regarding running time, since our proposed loss function inherently requires more computations than MSE loss, the running times of PPI and KM are higher than those of their corresponding PPI-loss and KM-loss. PPI's running time is higher than that of KM due to its frequent calls to the KM algorithm. GGPSO takes a significantly longer running time than PPI to achieve a comparable task completion rate, but it still performs significantly worse than PPI in rejection rate and average cost. In addition, when d is relatively small, the strong detour constraint would lead to an increase in the number of tasks being rejected. These rejected tasks will be carried over to the next task assignment batch (if they still have not expired at the next batch), where the algorithm will continue to search for suitable workers for these tasks. It can result in an accumulation of tasks across time batches, subsequently increasing the overall running time of the algorithms.

Effect of the number of spatial tasks. Fig. 6 describes the experimental results when varying the number of spatial tasks from $1K$ to $5K$. Overall, besides the best performance of PPI in the three quality metrics, the four metrics are all sensitive to the changes in the number of spatial tasks,



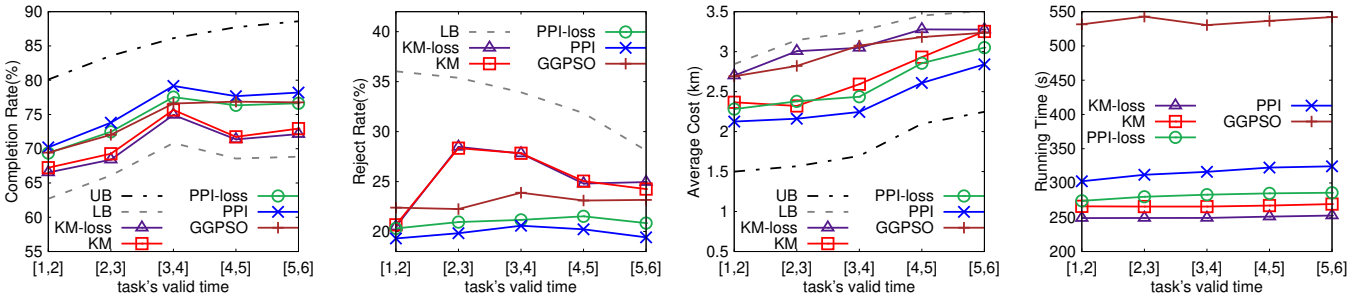
(a) Completion Rate (b) Rejection Rate (c) Average Cost (d) Running Time

Fig. 5. Effect of worker's detour d.



(a) Completion Rate (b) Rejection Rate (c) Average Cost (d) Running Time

Fig. 6. Effect of the number of spatial tasks



(a) Completion Rate (b) Rejection Rate (c) Average Cost (d) Running Time

Fig. 7. Effect of task's valid time

especially the task completion rate. As the number of spatial tasks increases, the task completion rate naturally shows a declining trend due to the limited number of workers. The rejection rate generally exhibits a trend of first increasing and then decreasing. When the number of tasks is relatively low, the algorithm assigns fewer tasks to workers, resulting in a lower probability of rejection. Conversely, when the number of tasks is high, as many workers are already occupied with tasks and do not require further assignment, the frequency of rejections slightly decreases. Similarly, when the number of tasks is large, the algorithm assigns tasks closer to workers to minimize the average cost for workers to complete tasks. This results in a decreasing trend in the average cost as the number of tasks increases from $4K$ to $5K$. Understandably, an increase in the number of tasks leads to a rise in the running time of all algorithms. Moreover, our proposed PPI and PPI-loss algorithms can achieve close performance to UB (e.g., when the number of spatial tasks is $4K$), further validating its effectiveness.

Effect of task's valid time. As shown in Fig. 7, when varying the valid time of task from [1, 2] to [5, 6], the task completion rate of all algorithms shows an upward trend. The reason is that a longer valid time for the task allows for a larger window in which it can be completed, resulting in a higher probability of completion. However, since the completion of the task is not only limited by its deadline but also influenced by the workers' tolerance for detours, the increase in completion rates is not very significant, and there is even a decline when the valid time changed from [3, 4] to [4, 5]. Regarding the rejection rate, only the KM and KM-loss algorithms show sensitivity to task's valid time. Overall, the algorithms with the lowest rejection rates are PPI and PPI-loss, followed by GGPSO, and lastly KM and KM-loss. Additionally, as the valid time increases, the average cost of completing the tasks shows an upward trend. The reason is that a longer valid time allows workers to complete tasks that are farther apart, as they have more time to reach the target locations of the tasks. The running time of the algorithms shows a slight increasing trend, as the

number of candidate tasks for workers increases. Of course, the running time of GGPSO is always significantly higher than that of other algorithms.

(3) Performance on Datasets Foursquar and Gowalla. In general, the performance of all the algorithms on datasets Foursquar and Gowalla is similar to that on datasets Porto and Didi. Please refer to Appendix C-B for details.

V. Related Work

In this section, we review the related works regarding predictive spatial crowdsourcing. In short, predictive spatial crowdsourcing means employing learning-based techniques to improve the efficiency and user experience of the spatial crowdsourcing platform.

Supply and Demand Prediction. In predictive-aware spatial crowdsourcing, it is crucial to understand the relationship between the demand of spatial tasks and the supply of crowd workers. For example, Cheng et al. [15] designed an effective grid-based prediction method to estimate the spatial distribution of future workers and tasks. Wang et al. [14] expected to achieve the highest total reward for task assignment by predicting worker churn. The authors in work [39] proposed a two-stage data-driven framework, where different learning models were used to predict the location and route of future workers in the prediction stage. Yuan et al. [40] designed a neural network model to predict future travel demand in various periods by considering regional-level correlation, time periodicity, and traffic correlation.

Task Completion Time Prediction. Recent efforts by academia and industry have explored pioneering solutions for predicting task completion time in spatial crowdsourcing [41]–[43]. For example, as for package delivery, Ruan et al. proposed a solution to infer delivery time based on the courier's trajectory automatically through three steps of data preprocessing, delivery location correction, and matching based on delivery events [41]. The MetaSTP model proposed by Ruan et al. [42] can accurately predict the last-mile delivery service time and performs well in handling complex delivery environments, location heterogeneity, and spatial observation bias. Yi et al. [43] proposed DeepSTA by designing an abnormal spatiotemporal learning module, using Node2vec to model road segment correlation, and employing LSTM models to capture workers' spatio-temporal dependencies.

Mobility Prediction. Mobility prediction is also a hot topic in predictive spatial crowdsourcing, which aims to establish models by analyzing workers' historical trajectory data for predicting workers' future mobility paths [8], [9], [44]. For example, Zhang et al. comprehensively utilized a fuzzy logic system to achieve accurate mobility predictions and designed a global heuristic search algorithm to optimize the overall task completion rate based on prediction results [8]. Ben et al. considered the participants' movement patterns and perception preferences, based on which they aimed to minimize the overall processing time of the perception task [44]. Wang et al. employed XGBoost to predict the pattern of riders choosing orders [9]. However, the above existing works have weaknesses in two aspects. On the one hand, they all ignored the cold start problem caused by the continuous arrival of new workers. On the other hand, they regard the prediction and task assignment stages as relatively separate, which weakens the guiding role of task assignment in training and the effective perception of prediction results in the task assignment stage.

Preference-aware Task Assignment. Preference-aware task assignment, especially with prediction-based preferences, is also related to our work. Indeed, the correlations and differences between preference-aware task assignment and TAMP depend mainly on how preferences and mobility are modeled and the aims of task assignment. In our work, worker's mobility is modeled as his/her trajectory. As such, when worker's preferences are modeled as the sequence of historical tasks they have completed in PATS [45], the prediction of worker's mobility and preferences are both sequence prediction problems, requiring similar sequence prediction techniques. However, in works [46], [47], preference is modeled as a value related to detour, reward, or other factors, completely different from mobility. On the other hand, while both preference-aware task assignment and TAMP aim to maximize task completion rates, their focal points differ. For instance, in work [47], the authors focus more on maximizing worker and task satisfaction, whereas our work emphasizes leveraging workers' predicted mobility to minimize task rejection rates.

VI. Conclusion

In this work, we study the task assignment problem in mobility prediction-aware spatial crowdsourcing to maximize the task completion rate, the task rejection rate, and the average cost for workers to complete tasks. To address this problem, we first introduce a worker-specific mobility prediction framework that leverages task-adaptive meta-learning represented as game theory-based learning task clustering and MAML-based meta-training. Then, to align the objectives of mobility prediction and task assignment, we devise a weighted loss function that incorporates historical spatial task data to modify the loss calculations. This ensures that the model is optimized for both prediction accuracy and its practical implications in task assignment. Furthermore, we explicitly quantify the impact of the mobility prediction model's performance on task assignment by designing a prediction performance-involved task assignment algorithm, which considers the reliability of mobility predictions, enabling more informed and effective task assignment. Finally, the experiment conducted on real-world datasets verified the effectiveness of our proposed methods.

Acknowledgment

This work is supported by the following grants: NSFC Grants 62372416, 61972362, and 62302460; HNSF Grant 242300421215; HK RGC Grants R1015-23 and 12202024. Corresponding author: Yafei Li.

REFERENCES

[1] 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," *Proc. VLDB Endow.*, vol. 7, no. 13, pp. 1629–1632, 2014.

[2] S. H. Kim, A. Alfarrarjeh, G. Constantinou, and C. Shahabi, "Tvdp: Translational visual data platform for smart cities," in *Proc. IEEE Int. Conf. Data Eng. Workshops*, pp. 45–52, 2019.

[3] G. Constantinou, O. Orhan, R. Kondepudi, H. Cho, S. H. Kim, A. Alfarrarjeh, and C. Shahabi, "Floravision: A spatial crowd-based learning system for california native plants," in *Proc. IEEE Int. Conf. Data Eng.*, pp. 2721–2724, 2021.

[4] "*OpenStreetMap*." https://www.openstreetmap.org.

[5] "*TaskRabbit*." https://www.taskrabbit.com.

[6] "*Waze*." http://www.waze.com.

[7] C. Chen, S.-F. Cheng, H. C. LAU, and A. MISRA, "Multi-agent task assignment for mobile crowdsourcing under trajectory uncertainties," in *Proc. Int. Conf. Auton. Agents Multi-Ag. Syst.*, pp. 1715–1716, 2015.

[8] J. Zhang and X. Zhang, "Multi-task allocation in mobile crowd sensing with mobility prediction," *IEEE Trans. Mob. Comput.*, vol. 22, no. 2, pp. 1081–1094, 2023.

[9] X. Wang, L. Wang, S. Wang, and et al., "Recommending-and-grabbing: A crowdsourcing-based order allocation pattern for on-demand food delivery," *IEEE Trans. Intell. Transp. Sys.*, vol. 24, no. 1, pp. 838–853, 2022.

[10] L. Wang, Z. Yu, Q. Han, and et al., "Multi-objective optimization based allocation of heterogeneous spatial crowdsourcing tasks," *IEEE Trans. Mob. Comput.*, vol. 17, no. 7, pp. 1637–1650, 2017.

[11] X. Zhu, Y. Luo, A. Liu, W. Tang, and M. Z. A. Bhuiyan, "A deep learning-based mobile crowdsensing scheme by predicting vehicle mobility," *IEEE Trans. Intell. Transp. Sys.*, vol. 22, no. 7, pp. 4648–4659, 2020.

[12] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. PMLR Int. Conf. Mach. Learn.*, pp. 1126–1135, 2017.

[13] Y. Liang and Z. Zhao, "Nettraj: A network-based vehicle trajectory prediction model with directional representation and spatiotemporal attention mechanisms," *IEEE Trans. Intell. Transp. Sys.*, vol. 23, no. 9, pp. 14470–14481, 2022.

[14] Z. Wang, Y. Zhao, X. Chen, and et al., "Task assignment with worker churn prediction in spatial crowdsourcing," in *Proc. ACM Int. Conf. Inform. Knowl. Manage.*, p. 2070–2079, 2021.

[15] P. Cheng, X. Lian, L. Chen, and C. Shahabi, "Prediction-based task assignment in spatial crowdsourcing," in *Proc. IEEE Int. Conf. Data Eng.*, pp. 997–1008, 2017.

[16] Y. Li, H. Li, X. Huang, J. Xu, Y. Han, and M. Xu, "Utility-aware dynamic ridesharing in spatial crowdsourcing," *IEEE Trans. Mob. Comput.*, vol. 23, no. 2, pp. 1066–1079, 2024.

[17] Y. Tong, Z. Zhou, Y. Zeng, and et al., "Spatial crowdsourcing: a survey," *VLDB J.*, vol. 29, no. 1, pp. 217–250, 2020.

[18] Y. Li, H. Li, B. Mei, X. Huang, J. Xu, and M. Xu, "Fairness-guaranteed task assignment for crowdsourced mobility services," *IEEE Trans. Mob. Comput.*, vol. 23, no. 5, pp. 5385–5400, 2024.

[19] M. Li, J. Wang, L. Zheng, H. Wu, P. Cheng, L. Chen, and X. Lin, "Privacy-preserving batch-based task assignment in spatial crowdsourcing with untrusted server," in *Proc. ACM Int. Conf. Inform. Knowl. Manage.*, pp. 947–956, 2021.

[20] Z. Liu, L. Liu, Y. Xie, and et al., "Task-adaptive meta-learning framework for advancing spatial generalizability," in *Proc. AAAI Conf. Artif. Intell.*, pp. 14365–14373, 2023.

[21] M. Lichman and P. Smyth, "Modeling human location data with mixtures of kernel densities," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, pp. 35–44, 2014.

[22] W. Liu, H. Lai, J. Wang, and et al., "Mix geographical information into local collaborative ranking for poi recommendation," *Springer World Wide Web*, vol. 23, pp. 131–152, 2020.

[23] A. Fallah, A. Mokhtari, and A. Ozdaglar, "Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach," in *Proc. Adv. Neural Inf. Process. Syst.*, pp. 3557–3568, 2020.

[24] H.-S. Park and C.-H. Jun, "A simple and fast algorithm for k-medoids clustering," *Expert Syst. Appl.*, vol. 36, no. 2, pp. 3336–3341, 2009.

[25] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder–decoder for statistical machine translation," in *Proc. Conf. Empir. Methods Nat. Lang. Process.*, pp. 1724–1734, 2014.

[26] A. Graves and A. Graves, "Long short-term memory," *Supervised sequence labelling with recurrent neural networks*, pp. 37–45, 2012.

[27] Y. Wang and S. Chen, "Multi-agent trajectory prediction with spatio-temporal sequence fusion," *IEEE Trans. Multimedia*, vol. 25, pp. 13–23, 2021.

[28] R. Yu and J. Sun, "Pose-transformed equivariant network for 3d point trajectory prediction," in *Proc. IEEE/CVF Conf. Comput. Vision Pattern Recognit.*, pp. 5503–5512, 2024.

[29] Y. Liu, Y. Bi, X. Yuan, D. Niyato, K. Yang, X. Chen, and L. Zhao, "A novel multimodal long-term trajectory prediction scheme for heterogeneous user behavior patterns," *IEEE Trans. Mob. Comput.*, 2024.

[30] B. Yang, G. Yan, P. Wang, C.-Y. Chan, X. Song, and Y. Chen, "A novel graph-based trajectory predictor with pseudo-oracle," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 33, no. 12, pp. 7064–7078, 2021.

[31] Y. Tong, D. Shi, Y. Xu, W. Lv, Z. Qin, and X. Tang, "Combinatorial optimization meets reinforcement learning: Effective taxi order dispatching at large-scale," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 10, pp. 9812–9823, 2021.

[32] X. Tang, F. Zhang, Z. Qin, Y. Wang, D. Shi, B. Song, Y. Tong, H. Zhu, and J. Ye, "Value function is all you need: A unified learning framework for ride hailing platforms," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, pp. 3605–3615, 2021.

[33] H. W. Kuhn, "The hungarian method for the assignment problem," *Nav. Res. Logist.*, vol. 2, no. 1-2, pp. 83–97, 1955.

[34] J. Munkres, "Algorithms for the assignment and transportation problems," *J. Soc. Ind. App. Math.*, vol. 5, no. 1, pp. 32–38, 1957.

[35] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and mobility: user movement in location-based social networks," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, pp. 1082–1090, 2011.

[36] "*Didi*." https://www.didiglobal.com.

[37] J. J. Levandoski, M. Sarwat, A. Eldawy, and M. F. Mokbel, "Lars: A location-aware recommender system," in *Proc. IEEE Int. Conf. Data Eng.*, pp. 450–461, 2012.

[38] D. Peng and S. J. Pan, "Clustered task-aware meta-learning by learning from learning paths," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2023.

[39] Y. Zhao, K. Zheng, Y. Cui, H. Su, F. Zhu, and X. Zhou, "Predictive task assignment in spatial crowdsourcing: a data-driven approach," in *Proc. IEEE Int. Conf. Data Eng.*, pp. 13–24, 2020.

[40] X. Qian, S. V. Ukkusuri, C. Yang, and F. Yan, "Short-term demand forecasting for on-demand mobility service," *IEEE Trans. Intell. Transp. Sys.*, vol. 23, no. 2, pp. 1019–1029, 2020.

[41] S. Ruan, Z. Xiong, C. Long, Y. Chen, J. Bao, T. He, R. Li, S. Wu, Z. Jiang, and Y. Zheng, "Doing in one go: delivery time inference based on couriers' trajectories," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, pp. 2813–2821, 2020.

[42] S. Ruan, C. Long, Z. Ma, J. Bao, T. He, R. Li, Y. Chen, S. Wu, and Y. Zheng, "Service time prediction for delivery tasks via spatial meta-learning," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, pp. 3829–3837, 2022.

[43] J. Yi, H. Yan, H. Wang, J. Yuan, and Y. Li, "Deepsta: A spatial-temporal attention network for logistics delivery timely rate prediction in anomaly conditions," in *Proc. ACM Int. Conf. Inform. Knowl. Manage.*, pp. 4916–4922, 2023.

[44] R. Ben Messaoud, Y. Ghamri-Doudane, and D. Botvich, "Preference and mobility-aware task assignment in participatory sensing," in *Proc. ACM Int. Conf. Model. Anal. Simul. Wireless Mob. Syst.*, pp. 93–101, 2016.

[45] Y. Zhao, K. Zheng, H. Yin, G. Liu, J. Fang, and X. Zhou, "Preference-aware task assignment in spatial crowdsourcing: from individuals to groups," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 7, pp. 3461–3477, 2020.

[46] Y. Zhao, J. Xia, G. Liu, H. Su, D. Lian, S. Shang, and K. Zheng, "Preference-aware task assignment in spatial crowdsourcing," in *Proc. AAAI Conf. Artif. Intell.*, pp. 2629–2636, 2019.

[47] X. Zhou, S. Liang, K. Li, Y. Gao, and K. Li, "Bilateral preference-aware task assignment in spatial crowdsourcing," in *Proc. IEEE Int. Conf. Data Eng.*, pp. 1687–1699, 2022.

[48] D. Monderer and L. S. Shapley, "Potential games," *Games Econ. Behav.*, vol. 14, no. 1, pp. 124–143, 1996.

[49] P. Cheng, L. Chen, and J. Ye, "Cooperation-aware task assignment in spatial crowdsourcing," in *Proc. IEEE Int. Conf. Data Eng.*, pp. 1442–1453, 2019.

[50] W. Meert and M. Verbeke, "Hmm with non-emitting states for map matching," in *Eur. Conf. Data Anal.*, 2018.
[51] P. Cheng, X. Lian, L. Chen, and C. Shahabi, "Prediction-based task assignment in spatial crowdsourcing," in *Proc. IEEE Int. Conf. Data Eng.*, pp. 997–1008, 2017.

APPENDIX A
PROOF OF THEOREMS AND LEMMAS

A. Proof of Theorem 1

Proof. According to [48], [49], for an exact potential game, a Nash equilibrium must exist that the best-response framework can find. As such, to prove Theorem 1, we only need to prove that $\mathcal{P}$ is an exact potential game. According to [48], a strategy game $\mathcal{P}$ is an exact potential game if there exists a potential function $F : \mathcal{ST} \rightarrow R$ such that $\forall i \in N$, for all $st_i \in ST_i$, it holds that $U_i(st_i', \vec{st}_{-i}) - U_i(st_i, \vec{st}_{-i}) = F_p(st_i', \vec{st}_{-i}) - F_p(st_i, \vec{st}_{-i})$ where st_i' and st_i are two strategies selected by player i, $\vec{st}_{-i}$ is the strategy sequence of other players except for i. The function F_p is an exact potential function for this strategy game. Consider the potential function $F_p(\vec{st}) = \sum_{\Gamma_i \in \vec{\Gamma}} U_i(\vec{st}) = \sum_{G \in \mathcal{G}} Q(G)$, which is the total utility of the players in W after they select strategies. $st_i^{'}$ and st_i represent that Γ_i choose to join into the learning task clusters G_j and G_k, respectively. We use $\mathcal{G}$ to represent the set of all the learning task clusters. As such, we can calculate that

$$
\begin{aligned}
&F_p(st_i^{'}, \vec{st}_{-i}) - F_p(st_i, \vec{st}_{-i}) \\
&= \sum_{\Gamma_i \in \vec{\Gamma}} U_i(st_i^{'}, \vec{st}_{-i}) - \sum_{\Gamma_i \in \vec{\Gamma}} U_i(st_i, \vec{st}_{-i}) \\
&= \big(Q(G_j) + Q(G_k \setminus \{\Gamma_i\}) + \sum_{G \in \mathcal{G} \setminus \{G_i, G_k\}} Q(G)\big) \\
&- \big(Q(G_j \setminus \{\Gamma_i\}) + Q(G_k) + \sum_{G \in \mathcal{G} \setminus \{G_i, G_k\}} Q(G)\big) \\
&= \big(Q(G_j) - Q(G_j \setminus \{\Gamma_i\})\big) - \big(Q(G_k) - Q(G_k \setminus \{\Gamma_i\})\big) \\
&= u(\Gamma_i, G_j) - u(\Gamma_i, G_k) \\
&= U_i(st_i^{'}, \vec{st}_{-i}) - U_i(st_i, \vec{st}_{-i}).
\end{aligned}
$$

Therefore, we can conclude that the n-player strategy game $\mathcal{P}$ is an exact potential game, and Theorem 1 is proved. □

B. Proof of Lemma 1

Proof. We prove Lemma 1 according to the example in Fig 4. To prove that the worker's detour constraint is met, we only need to prove that $dis(l_1, \tau.l) + dis(\tau.l, l_2) - dis(l_1, l_2) < d$. Since $dis(l_1, \hat{l}_1) \leq a$, $dis(\hat{l}_i, \tau.l) \leq b$, and $a+b \leq d/2$, it holds that $dis(l_1, \tau.l) \leq a + b \leq d/2$. Then it is straightforward to deduce from the triangle inequality that $dis(\tau.l, l_2) < dis(l_1, \tau.l) + dis(l_1, l_2)$. As such, it holds

$$
\begin{aligned}
&- dis(l_1, \tau.l) < dis(l_1, l_2) - dis(\tau.l, l_2) \\
&\& \ 2 * dis(l_1, \tau.l) \leq d \\
&\Rightarrow dis(l_1, \tau.l) < d + dis(l_1, l_2) - dis(\tau.l, l_2) \\
&\Rightarrow dis(l_1, \tau.l) + dis(\tau.l, l_2) - dis(l_1, l_2) < d.
\end{aligned}
$$

Therefore, if $dis(l_1, \hat{l}_1) \leq a$ with probability $MR(r, \hat{r})$, we can draw the conclusion that w can complete τ without violating his/her detour constraint with probability $MR(r, \hat{r})$ when $a + b \leq d/2$. Lemma 1 is proved. □

C. Proof of Lemma 2

Proof. Since $dis(l_1, \tau.l) \leq a + b$ with the probability $MR(r, \hat{r})$, it holds $dis(l_1, \tau.l) < sp * (\tau.t - t_c)$ with the same probability. Then, $dis(l_1, \tau.l)/sp + t_c < \tau.t$, i.e., the deadline constraint of task τ is met with the probability $MR(r, \hat{r})$. Lemma 2 is proved. □

D. Proof of Theorem 2

Proof. The proof can be straightforwardly derived by combining Lemma 1 and Lemma 2. Thus, it is omitted. □

APPENDIX B
COMPLEXITY ANALYSIS OF ALGORITHMS

A. Complexity Analysis of Algorithm 1

Algorithm 1 clusters all learning tasks into a learning task tree of $|F_s| + 1$ layer. It takes $O(lk|G|d_s + B|G||ST|d_s)$ time to cluster G at layer s into k sub-clusters, where $O(lk|G|d_s)$ is the time complexity of k-medoids algorithm, $O(B|G||ST|d_s)$ is the time complexity of best-response framework, k is the number of sub-clusters, l is the iterations in k-medoids algorithm, B is the number of best response strategy selection iterations, $|ST|$ is the average number of strategies of learning tasks in G, and d_s is the data dimension of the clustering factors at layer s. Then, for all the clusters in layer s, it takes $O(lk|N|d_s + B|N||ST|d_s)$, where $|N|$ is the total number of learning tasks. As such, the time complexity of algorithm 1 is $O(|N|d(lk + B|ST|)$, where d is the total dimension of all the clustering factors. Since only leaf nodes store data at any time, the space complexity of Algorithm 1 is $O(|N|d)$.

B. Complexity Analysis of Algorithm 2

For each leaf node in the learning task tree T^t, it takes time $O(l(mkp^r|\Gamma^s| + |\Gamma^q|p^r) + (mkp^g + p^g))$ to train it (i.e., the time complexity of algorithm 3). For each non-leaf node, it takes $O(p^g)$ time to update the parameter and $|T^t.\mathcal{CH}|$ (i.e., the number of child nodes) time to sum the losses. As such, the time complexity of algorithm 2 is $O(n_l^g(l(mkp^r|\Gamma^s| + |\Gamma^q|p^r) + (mkp^g + p^g)) + n_n^g p^g + n^e)$, where n_l^g, n_n^g, and n^e is the number of leaf nodes, non-leaf nodes and edges in T^t. Since algorithm 2 is recursive, it needs to store all the parameters and gradients corresponding to each node. The space complexity of algorithm 2 is $O(p^g(n_l^g + n_n^g))$.

C. Complexity Analysis of Algorithm 3

For each training iteration in algorithm 3, it takes $O(mkp^r|\Gamma^s| + |\Gamma^q|p^r)$ time to calculate the adapt loss and query loss (line 6 and line 8) and $O(mkp^g + p^g)$ time to update the parameters (line 7 and line 9), where p^r is the time complexity of a single inference in the neural network and p^g is the time complexity of parameter updates (i.e., $\theta \in R^{p^g}$). Γ^s and Γ^q are the instances in the support and query sets of the sampled learning task Γ. As such, the time complexity of algorithm 3 is $O(l(mkp^r|\Gamma^s| + |\Gamma^q|p^r) + (mkp^g + p^g))$. It only

TABLE V
STATISTICS OF EXPERIMENTAL DATASETS

Dataset	Simulation Object	Size
Porto	Worker	442
Didi	Task	7,065,937
Gowalla	Worker	196,591
Foursquare	Task	1,021,970

stores all the instances of the learning task cluster in $T^t.G$ and the parameter $T^t.\theta$. In each iteration, it only stores parameter $\theta_i \in R^{p^g}$, gradient $\nabla \mathcal{L}^s_{T_i}(f_{\theta_i}) \in R^{p^g}$ and $\nabla \mathcal{L}^q_{T_i}(f_{\theta_i}) \in R^{p^g}$. As such, the space complexity of algorithm 3 is $O(|G| + p^g)$.

APPENDIX C
SUPPLEMENTARY EXPERIMENTS

A. Datasets and Preprocessing Details

As shown in table V, Porto[2] (Gowalla [35]) and Didi [36] (Foursquare [37]) datasets are used to simulate crowd workers and spatial tasks, respectively. Specifically, Porto dataset includes the travel trajectories of 442 taxis in Porto from July 1, 2013 to June 30, 2014. First, we employed an HMM-based map matching algorithm [50] to match taxi trajectories to the urban road network, in order to eliminate noise in the trajectory data (such as unreasonable points that are not on the roads). Second, to enhance the feasibility of the number of workers within a single day, we transformed the data, retaining the temporal distribution of trajectories across the day, and mapped it to 10 days spanning from October 20 to October 30, 2013. This approach allowed for augmentation in daily worker count while maintaining the integrity of the original trajectory patterns. The data of 8 days from October 20 to October 28 is used as the training dataset, and the data of 2 days from October 29 to October 30 is regarded as the test dataset. Third, we divide the main area where the trajectories are located into 100×50 grids, where each grid is represented as a 2-tuple $(latitude_i, longitude_j)(1 \leq latitude_i \leq 100, 1 \leq latitude_j \leq 50)$. Then, we map the original latitude and longitude to the corresponding grid indexes in the longitude and latitude directions. We obtained POI information for representing spatial features from OpenStreetMap [4]. Regarding spatial tasks, we generate according to a real-world ride-hailing order dataset released by Didi Chuxing from Chengdu, which includes all the orders from November 1 to November 30, 2016. We mapped the original locations of the orders in Chengdu to the location in Porto linearly and the pick-up location of each order was mapped to the target location of the spatial task. The arrival pattern of spatial tasks was simulated by the arrival pattern of orders in Didi and the deadline of each spatial task was randomly generated according to the validity period. Following work [51], we also employed datasets Gowalla and Foursquare to evaluate our proposed approaches. Similarly, they are processed in the same way as Porto and Didi.

[2]https://www.kaggle.com/c/pkdd-15-predict-taxi-service-trajectory-i/overview

TABLE VI
EFFECT OF LEARNING TASK CLUSTERING ALGORITHM AND FACTORS ON DATASET GOWALLA

cluster algorithm	cluster factor			metric			
	Sim_d	Sim_s	Sim_l	RMSE	MAE	MR	TT
GTMC	✓	✗	✗	0.3843	0.3460	0.8441	3002.2
	✗	✓	✗	0.3991	0.3613	0.8368	2713.8
	✗	✗	✓	0.3893	0.3513	0.8486	2773.0
	✓	✓	✗	0.3827	0.3466	0.8464	5256.9
	✓	✓	✓	0.3756	0.3410	0.8591	5683.0
k-means	✓	✗	✗	0.3839	0.3464	0.8352	3231.0
	✗	✓	✗	0.4086	0.3700	0.8393	3258.5
	✗	✗	✓	0.3907	0.3508	0.8414	3356.7
	✓	✓	✗	0.3898	0.3497	0.8381	4861.0
	✓	✓	✓	0.3883	0.3487	0.8406	6103.6

B. Experiments on Datasets Foursquare and Gowalla

In this section, we analyse the experimental results on datasets Foursquare and Gowalla in detail.

(1) Performance of Mobility Prediction. First, we present the experimental results of mobility prediction performance on datasets Foursquare and Gowalla.

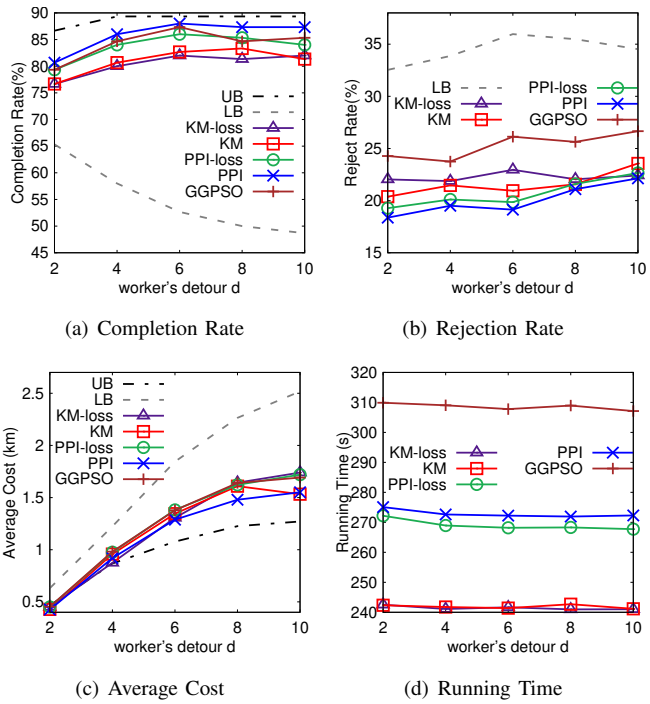
Ablation Study on Clustering Algorithms and Factors. As shown in table VI, when distribution feature, spatial feature, and learning path feature are considered separately as the clustering factor, distribution feature is the most effective one, followed by learning path feature and spatial feature. Compared with the experiments on the Porto dataset, it indicates that the impact of spatial feature on the mobility prediction task in the dataset is relatively small. Similarly, considering more factors in the clustering of learning tasks achieves a better performance than using single factors. Moreover, in all cases, GTMC outperforms the k-means algorithm in terms of RMSE, MAE, and MR, further verifying the effectiveness of our proposed game theory-based algorithm. Regarding TT, it shares the same pattern with that on dataset Porto.

Effect of seq_{in} and seq_{out}. Table VII presents the performance of four algorithms MAML, CTML, GTTAML-GT, and GTTAML on four metrics RMSE, MAE, MR, and TT, when varying the input sequence length seq_{in} and output sequence seq_{out} for mobility prediction. Similar to experimental results on dataset Porto, our GTTAML displays the best performance in RMSE, MAE, and MR, followed by GTTAML-GT, CTML and MAML. As expected, the performance of four algorithms exhibits a trend of improvement with seq_{in} increases. Additionally, when varying seq_{out}, the performance of all four algorithms shows a decreasing trend, but GTTAML always maintains the best performance. The training time spent by algorithms GTTAML-GT and GTTAML is the highest because the learning task trees constructed by the two algorithms have more nodes. However, since the process of training is offline, such training time is tolerable.

(2) Performance of Task Assignment. We illustrate the experimental results on datasets Foursquare and Gowalla to verify the effectiveness of our proposed task assignment algorithm.

TABLE VII
EFFECT OF seq_{in} AND seq_{out} ON DATASET GOWALLA

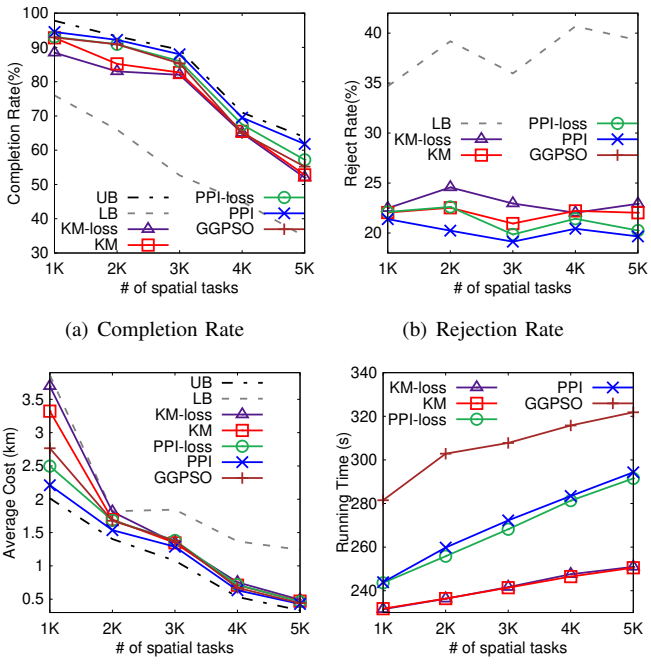
seq_{in}	metric	MAML	CTML	GTTAML-GT	GTTAML
1	RMSE	0.4202	0.3908	0.3883	**0.3756**
	MAE	0.3773	0.3518	0.3487	**0.3410**
	MR	0.8369	0.8387	0.8406	**0.8591**
	TT	3792.1	5045.0	6124.0	5683.0
5	RMSE	0.4430	0.4053	0.3984	**0.3932**
	MAE	0.3998	0.3605	0.3551	**0.3518**
	MR	0.8145	0.8347	0.8386	**0.8457**
	TT	4336.4	6106.3	6863.5	6630.2
10	RMSE	0.5478	0.5401	0.4728	**0.4340**
	MAE	0.4909	0.4798	0.4200	**0.3812**
	MR	0.8298	0.8272	0.8324	**0.8430**
	TT	7280.0	7966.3	8656.9	8170.3
seq_{out}	metric	MAML	CTML	GTTAML-GT	GTTAML
1	RMSE	0.3733	0.3724	0.3577	**0.3387**
	MAE	0.3365	0.3345	0.3218	**0.3041**
	MR	0.8410	0.8385	0.8447	**0.8630**
	TT	2511.7	2941.2	3066.0	3313.3
2	RMSE	0.4202	0.3908	0.3883	**0.3756**
	MAE	0.3773	0.3518	0.3487	**0.3410**
	MR	0.8369	0.8387	0.8406	**0.8591**
	TT	3792.1	5045.0	6124.0	5683.0
3	RMSE	0.4342	0.3805	0.3760	**0.3625**
	MAE	0.3911	0.3432	0.3388	**0.3272**
	MR	0.8348	0.8416	0.8387	**0.8500**
	TT	5591.3	7876.2	8970.0	8378.9



(a) Completion Rate (b) Rejection Rate

(c) Average Cost (d) Running Time

Fig. 8. Effect of worker's detour d on datasets Foursquare and Gowalla.



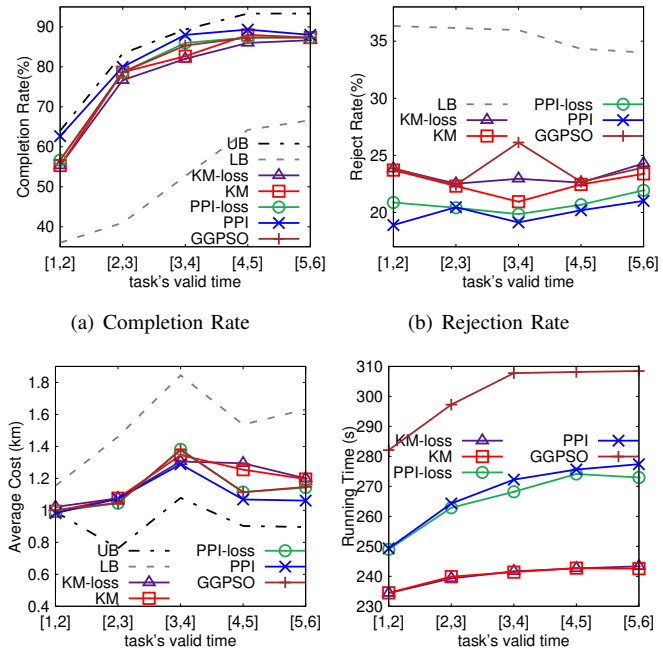
(a) Completion Rate (b) Rejection Rate

(c) Average Cost (d) Running Time

Fig. 9. Effect of the number of spatial tasks on datasets Foursquare and Gowalla.

Effect of worker's detour d**.** As illustrated in Fig. 8, we varied the detours that workers can tolerate to verify the effectiveness of the algorithms on datasets Foursquare and Gowalla. Overall, the experimental results on datasets Foursquare and Gowalla show a similar pattern with those on datasets Porto and Didi. The PPI algorithm still performs best in the three metrics: completion rate, rejection rate, and average cost. Compared to GGPOS, PPI achieved a higher task completion rate, a lower task rejection rate, and a smaller average cost with a shorter running time. However, in contrast to the experimental results on datasets Porto and Didi, the differences in average cost among different algorithms are smaller. This may be related to the more similar data distributions in the task dataset Foursquare and the worker dataset Gowalla. At the same time, this is the main reason why, in the experiments on datasets Foursquare and Gowalla, the rate of decrease in task rejection and the rate of increase in task completion are both slower as the workers' tolerance for detours increases.

Effect of the number of spatial tasks. Fig. 9 shows the experimental results when we varied the number of spatial tasks. Similar to results on datasets Porto and Didi, the completion rate decreases as the number of spatial tasks increases. Also, the running time shows an increasing trend, which is reasonable since more spatial tasks require more time to handle. The task rejection rate remains insensitive to changes in the number of spatial tasks, as it is primarily influenced by the performance of the mobility prediction model. However, the average cost of all algorithms significantly decreases as the number of tasks increases. This is because, when there are more spatial tasks, workers can choose tasks that are closer to them. Regardless of how the number of tasks changes, PPI can always perform best in the three metrics: completion rate, rejection rate, and average cost.



(a) Completion Rate (b) Rejection Rate

(c) Average Cost (d) Running Time

Fig. 10. Effect of task's valid time on datasets Foursquare and Gowalla.

Effect of task's valid time. Fig. 10 describes the experimental results on datasets Foursquare and Gowalla, when varying the valid time of spatial tasks. In general, the task completion rate is most sensitive to changes in the task's valid time, showing a clear increasing trend. In contrast, the change in the rejection rate exhibits a fluctuating trend, and this trend varies among different algorithms. Regarding average cost, similar to the other two groups of experiments, the differences in average costs among different algorithms are relatively small, generally showing a trend of increasing. The running time of all algorithms increases with the valid time of the tasks, but the speed of increase gradually slows down. This is because the increase in task's valid time provides workers with more candidate tasks to evaluate. However, when task's valid time reaches a certain level, it no longer serves as the primary constraint limiting the number of candidate tasks for the workers, resulting in a gradual slowdown in the rate of increase.