Spatial Meta Learning with Comprehensive Prior Knowledge Injection for Service Time Prediction

Shuliang Wang, Qianyu Yang, Sijie Ruan, Cheng Long, Ye Yuan,
Qi Li, Ziqiang Yuan, Jie Bao and Yu Zheng, *Fellow, IEEE*

Abstract—Intelligent logistics relies on accurately predicting the service time, which is a part of time cost in the last-mile delivery. However, service time prediction (STP) is non-trivial given complex delivery circumstances, location heterogeneity, and skewed observations in space, which are not well-handled by existing solutions. In our prior work, we treat STP at each location as a learning task to keep the location heterogeneity, propose a prior knowledge-enhanced meta-learning to tackle skewed observations, and introduce a Transformer-based representation module to encode complex delivery circumstances. Maintaining the design principles of prior work, in this extended paper, we propose MetaSTP⁺. In addition to fusing the prior knowledge after the meta-learning process, MetaSTP⁺ also injects the prior knowledge before and during the meta-learning process to better tackle skewed observations. More specifically, MetaSTP⁺ completes the support set of tasks with scarce samples from other tasks based on prior knowledge and is equipped with a prior knowledge-aware historical observation encoding module to achieve those purposes accordingly. Experiments show MetaSTP⁺ outperforms the best baseline by 11.2% and 8.4% on two real-world datasets. Finally, an intelligent waybill assignment system based on MetaSTP⁺ is deployed in JD Logistics.

Index Terms—Delivery Data Mining, Meta-Learning, Urban Computing.

1 Introduction

The last-mile delivery in logistics mainly relies on couriers. Typically, once a batch of parcels arrives at a delivery station, it would be assigned to couriers. Then, couriers would start a delivery trip to deliver the assigned parcels at several locations, as shown in Figure 1(a). Two types of time cost for the entire trip are involved in the delivery trip: *travel time* and *service time*. The former is the time cost traveling between locations, and the later is the time cost completing the delivery for a set of parcels at a certain location, namely, a *delivery task*. Estimating those two types of time cost facilitates many downstream applications, e.g., the route planning with time windows [1], [2], the workload balancing [3] and the delivery time prediction [4] as demonstrated in Figure 1(b). While the travel time prediction has been widely studied [5], [6], [7], there are limited research focusing on the service time prediction so far. Therefore, in this work, we mainly focus on the service time prediction.

The challenges of the service time prediction problem are from three aspects: 1) *Complex delivery circumstances.* Figure 2 shows a building with two units under three delivery circumstances, which have the same number of customers to be delivered to, but totally different delivery processes, leading to different service time. 2) *Location heterogeneity.* The service time could also vary greatly for deliveries at different locations. As observed in Figure 2(d), the service time could vary for different types or structure of buildings. 3) *Skewed observations.* The historical delivery tasks, i.e., data samples for training, distributed highly imbalanced among locations, as shown by the heat map in Figure 2(d).

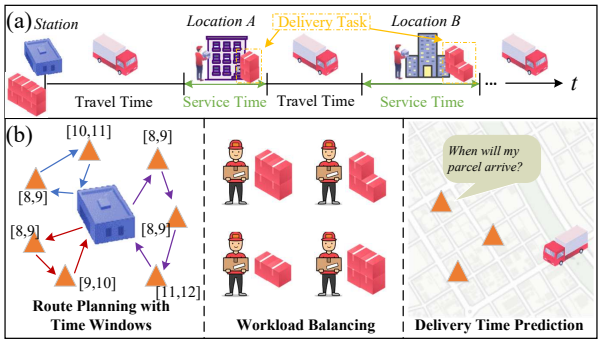
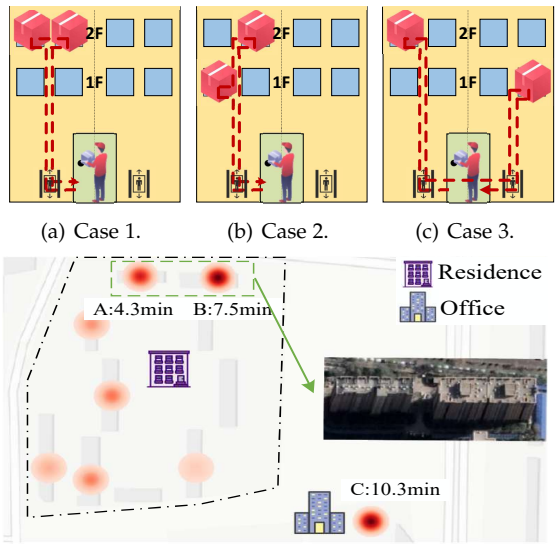


Fig. 1. Service time and its applications.

Inspired by techniques of Transformer to encode the correlation among fine-grained elements and meta-learning to learn from small samples, in our prior work [8], we present MetaSTP, a Meta-learning-based method to make the Service Time Prediction. To tackle the location heterogeneity, MetaSTP treats the service time prediction at different locations as different *learning tasks*. When predicting the service time, MetaSTP first leverages a Transformer encoder-based representation module to obtain floor-distribution-aware delivery task embedding, which captures the complex delivery circumstances of each delivery task, then employs a location prior knowledge enhanced meta-learning method to produce accurate predictions based on location-

- *Shuliang Wang, Qianyu Yang, Sijie Ruan, Ye Yuan and Ziqiang Yuan are with the School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China. E-mail: {slwang2011,yangqy,sjruan,yuanye,ziqiangy}@bit.edu.cn*
- *Cheng Long is with the College of Computing and Data Science, Nanyang Technological University, Singapore. E-mail: c.long@ntu.edu.sg*
- *Qi Li is with the School of Information Science and Technology, Beijing Forestry University, Beijing, China. E-mail: liqi2024@bjfu.edu.cn*
- *Jie Bao and Yu Zheng are with JD iCity, JD Technology, Beijing, China and JD Intelligent Cities Research, China. E-mail: baojie@jd.com, msyuzheng@outlook.com*

Manuscript received 27 Jan. 2024; revised 17 Oct. 2024; accepted 26 Nov. 2024.
(Corresponding author: Sijie Ruan.)



(a) Case 1. (b) Case 2. (c) Case 3.

(d) Location heterogeneity & Skewed observations.

Fig. 2. Challenges of service time prediction.

specific historical observations, globally learned delivery knowledge and location prior knowledge, to mitigate the skewed observation issue.

Although MetaSTP carefully tackled the challenges of complex delivery circumstances and the location heterogeneity thanks to fine-grained delivery task representation and the individual location modeling, the location prior knowledge enhanced meta-learning can be improved to tackle the challenge of skewed observations. Because in MetaSTP, the utilization of the prior knowledge is a bit shallow, which is only injected after the meta-learning process. We identify two limitations of this design:

- **Inadequate modeling for locations with extremely scarce historical observations.** Even though meta-learning is capable to learn from few-shot data, the sizes of support sets for a considerable proportion of locations are still too small to achieve accurate prediction, which cannot be fully compensated via the late fusion of the location prior knowledge. As shown in Figure 3, though location C has moderate training samples compared with location A, it can still achieve low prediction error as location A did, benefiting from the knowledge sharing mechanism of meta-learning. However, the sizes of support set for location B, E and D are too small to model the corresponding delivery events well.

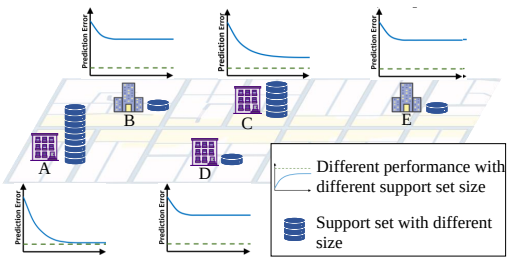


Fig. 3. Locations with extremely scarce historical observations have modeling difficulties even meta-learning is employed.

- **Ignorance of the prior knowledge when calculating sample correlations.** When modeling the correlation among the query sample and samples in the support set, the prior knowledge is ignored. However, the importance of each historical observation to the query sample depends on the location. For example, as shown in Figure 4, assume we have a support sample and a query sample at a certain location. The support sample contains parcels to be delivered to floor 6, while the query sample contains parcels to be delivered to floor 2. If we ignore the prior knowledge when calculating the sample correlation, the correlation calculation module would produce the same correlation score no matter the building is equipped with elevators or not, e.g., 0.5, as the top part of Figure 4 shown. If the prior knowledge is incorporated, we can achieve prior knowledge-aware correlation calculation. As the bottom part of Figure 4 shown, their correlations are 0.3 and 0.8 for buildings without/with elevator, respectively, which fits the cases of real world well.

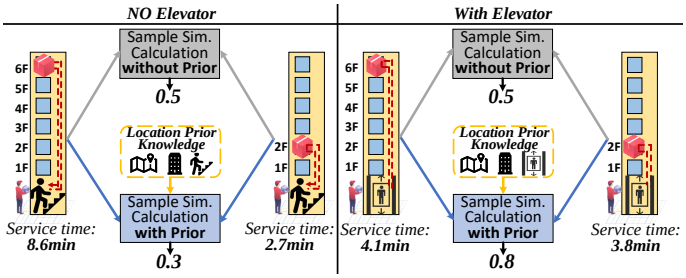


Fig. 4. The prior knowledge affects the sample correlation calculation.

Maintaining the design principles of MetaSTP, we propose MetaSTP$^+$, which injects the location prior knowledge more comprehensive than MetaSTP to tackle the skewed observations better: in addition to fusing the location prior knowledge *after* historical observations are encoded, we inject the location prior knowledge *before* and *during* the meta-learning process in different ways to overcome aforementioned limitations. More specifically, *to address the limitation of inadequate modeling for locations with extremely scarce historical observations*, we complete the support set of a learning task with scarce samples with those from similar learning tasks, the similarities of which are derived from a pre-trained service time-guided location embedding module. *To address the limitation of ignorance of the prior knowledge when calculating sample correlations*, we devise a novel prior knowledge-aware historical observation encoding module to consider the prior knowledge of locations in the process.

Our contributions are four folds:

1) We identify major challenges of service time prediction, and propose a meta-learning-based service time prediction model, i.e., MetaSTP$^+$. It fuses the prior knowledge of tasks before, during and after the meta-learning process, which is the first of this kind.
2) We design a location similarity-aware support set completion strategy to handle tasks with extremely scarce historical observations.
3) We propose a prior knowledge-aware historical observation encoding module to better model the correlation between the query sample and the support set.
4) Extensive experiments based on two real delivery datasets from JD Logistics demonstrate the effectiveness of MetaSTP$^+$, which outperforms the best baseline by 11.2% and 8.4%, respectively. Besides, we present an in-

telligent waybill assignment system based on MetaSTP+, which has been deployed and used internally in JD Logistics.

Among the contribution items, 2) and 3) are completely newly introduced in this extension, and the others are partially covered in our prior work.

2 PRELIMINARIES

2.1 Problem Formulation

Definition 1 (Waybill). A waybill contains the information about the parcel to deliver, denoted as a 4-tuple $w = (addr, uid, \mathbf{F}, ts)$. $addr$ is the address, uid is the customer ID, and $\mathbf{F}$ denotes the features of the corresponding parcel, e.g., the weight and the volume. ts is a planned delivery time slot of the parcel (e.g., pending to be delivered during 8AM-11AM), which depends on how many delivery trips would be conducted in a day.

Definition 2 (Delivery Task). The set of parcels to be delivered together to the same delivery location is named as a delivery task. Parcels in a delivery task share the same planned delivery time slot, therefore, we explicitly denote a delivery task as $t = (W, ts)$, where W is the set of waybills of those parcels.

Note that, the delivery location of each waybill, is a geospatial coordinate in the urban space, which can be obtained based on its shipping address via Geocoding [1], or couriers' annotation [9], [10].

Problem Definition. Given historical delivery tasks and their corresponding service time, the service time prediction (STP) problem is to predict the service time for a delivery task in the future.

We did not consider identities of couriers, since they are actually implicitly captured because a courier is responsible for a delivery zone within a period of time. Another advantage is that the generalization ability of our model would be better, e.g., we can predict the service time for new couriers. This choice is also employed by the existing work [11].

2.2 Basic Concepts of Meta-Learning

Meta-learning [12] aims to extract meta-knowledge that is globally shared among related *learning tasks*, enabling effective predictions even with limited observations. Each learning task $\mathcal{T}$ comprises a support set $\mathcal{D}^s$ and a query set $\mathcal{D}^q$. The inference for a query sample $\boldsymbol{x}^q$ is formulated as:

$$\hat{y}^q = f_{\boldsymbol{\theta}}(\boldsymbol{x}^q, \mathcal{D}^s) \tag{1}$$

where $f_{\boldsymbol{\theta}}$ is a neural network parameterized by the meta-knowledge $\boldsymbol{\theta}$. To optimize $\boldsymbol{\theta}$, we use a set of meta-training tasks $\mathscr{T}_{\text{meta-train}}$ sampled from the task distribution $p(\mathcal{T})$. The meta-loss function is:

$$\mathscr{L}(\boldsymbol{\theta}) = \sum_{\mathcal{T}_i \in \mathscr{T}_{meta-train}} \frac{1}{|\mathcal{D}_i^q|} \sum_{(\boldsymbol{x}^q, y^q) \in \mathcal{D}_i^q} \mathcal{L}(f_{\boldsymbol{\theta}}(\boldsymbol{x}^q, \mathcal{D}_i^s), y^q) \tag{2}$$

where $\mathcal{L}$ is the loss function for the learning tasks. The objective is to find $\boldsymbol{\theta}$ that generalizes well to new tasks sampled from $p(\mathcal{T})$ based on these formulations. We mainly follow the paradigm of the model-based meta-learning because it is easier to be optimized [13].

1. https://en.wikipedia.org/wiki/Geocoding

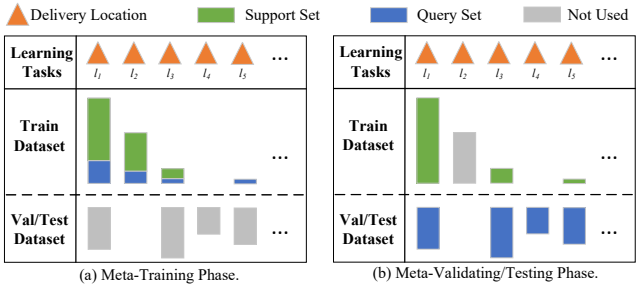


Fig. 5. Illustration of learning task setup.

3 METHODOLOGY

3.1 Learning Task Setup

We leverage fixed timestamps to split historical delivery tasks into training dataset, validation dataset and testing dataset (same as traditional machine learning settings). We treat delivery tasks at each location as a learning task.

For each location in the meta-training dataset $\mathscr{T}_{meta-train}$, we always keep $r\%$ proportion of data as the query set (in blue) and leave others as the support set (in green). And for $\mathscr{T}_{meta-val}$/$\mathscr{T}_{meta-val}$, the whole training dataset of each location is treated as the support set, and the whole validation/test dataset is treated as the query set, as shown in Figure 5(b).

Some locations have no delivery record during the time interval of the training dataset, resulting in empty support set during the meta-validating/testing, as l_4 shown in Figure 5(b). Given this fact, we also include those learning tasks with empty support set in the meta-training, e.g., l_5 in Figure 5(a). It is achieved by specifying a minimum query set threshold N_q^{min}. For a location in the meta-training, if the size of query portion is less than N_q^{min}, all samples would be assigned to the query set during the meta-training, which is formally given as follows:

$$N_q = \begin{cases} \lceil r\% * |\mathcal{D}| \rceil, & \text{if } \lceil r\% * |\mathcal{D}| \rceil \geq N_q^{min} \\ |\mathcal{D}|, & \text{otherwise} \end{cases} \tag{3}$$

where $\mathcal{D}$ is the whole training dataset of a location.

3.2 Model Overview

After we setup the learning tasks by location, the location heterogeneity can be captured. However, skewed observations and complex delivery circumstances should be further handled. To tackle the former challenge, the main idea is to use samples in the support set of similar learning tasks to complete the support set of the learning task with scarce observations, and fuse the prior knowledge of each delivery location in the task adaption process. To tackle the later challenge, we further introduce a fine-grained representation layer in the prediction model.

Since the spatial features are available even if there are no delivery events ever occurred at a location, we use them as the prior knowledge of each learning task. For example, the locations with the same POI type are more likely to have similar building structures, thus their delivery situations may be similar. However, the raw spatial features might be sparse and noisy, we first transform them into a dense representation space, and use the latent embedding to complete the support set and help the later service time prediction.

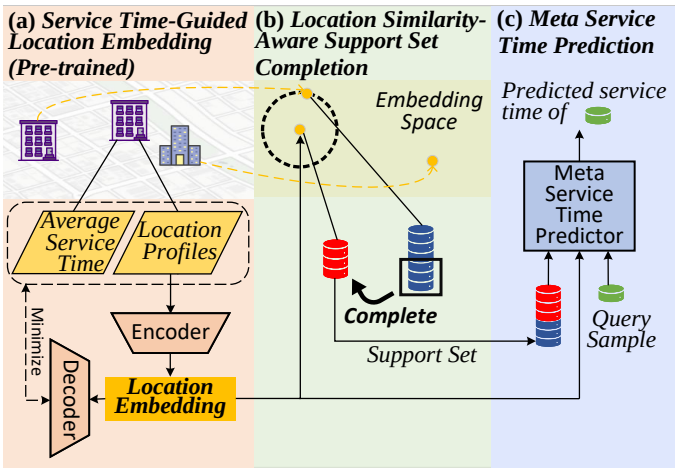


Fig. 6. The main idea of MetaSTP$^+$.

Figure 6 depicts the overview of MetaSTP$^+$, which consists of 3 stages to predict the service time for each delivery task:

- **Service Time-Guided Location Embedding**, which learns the latent representation of each location based on the location profiles via an auto-encoder. During the pre-training process, we further introduce the auxiliary task, i.e., average service time prediction, to make sure the learnt location embeddings are suitable for the service time prediction (Figure 6(a)).
- **Location Similarity-Aware Support Set Completion**, which completes the support set of learning tasks with scarce samples using that of similar tasks, the similarities of which are characterized by the previously learnt location embeddings (Figure 6(b)).
- **Meta Service Time Prediction**, which predicts the service time of each query sample in a learning task based on delivery task features, corresponding (completed) support set as well as the learnt location embedding (Figure 6(c)).

The detailed architecture of MetaSTP$^+$ is shown in Figure 7. Next, we elaborate each stage in detail.

3.3 Service Time-Guided Location Embedding

The module aims to learn the prior knowledge of each learning task to help the later support set completion and the prediction.

Main Idea. Since raw location profiles are sparse and noisy, we need to transform them into dense representations. There exist various existing works to learn embeddings of locations based on multi-modal data [14], [15], [16]. However, given that multi-model data may not be always available on delivery locations, we seek for embedding models that can take simple semantic features of locations. However, embeddings learned solely through self-supervision (e.g., auto-encoders) may not enhance service time prediction. Therefore, we augment the training with an auxiliary task that predicts the average service time of each location based on its embedding. This ensures the learned embeddings are not only dense and informative but also strongly correlated with service time prediction.

Implementation. Figure 7(a) depicts the Service Time-Guided Location Embedding module, which follows the encoder-decoder structure to learn the prior knowledge of each location, i.e., location embedding. After the module is trained, only the encoder is reserved to produce the prior knowledge of each location.

We consider four types of location profiles as the input: (1) region (row and column index of a $500m \times 500m$ cell in the gridded urban space), (2) POI type, (3) built year, and (4) second-hand house price per square meter. The last two features implicitly reflect whether the building is equipped with the elevator, which is usually not publicly available but also a very important factor to affect the service time.

The encoder transforms the POI type into a one-hot vector $\mathbf{x}_{poi}$, embeds it through an embedding layer, concatenates it with other quantitative features $\mathbf{x}_{quant}$, and processes them through a four-layer feedforward network with LeakyReLU activations to obtain the location prior knowledge representation $\mathbf{l}_{emb}$.

The decoder aims to perform the location profile reconstruction and the average service time prediction, it uses the learned embedding $\mathbf{l}_{\text{emb}}$ as input, and consists of three components, which use different feedforward networks (with the same number of layers as the encoder does):

(1) Quantitative Profile Reconstructor, reconstructing quantitative features $\hat{\mathbf{x}}_{\text{quant}}$.(2) POI Classifier, infering the POI type distribution $\hat{\mathbf{p}}_{\text{poi}}$ of a location using an additional softmax layer before the output.(3) Average Service Time Predictor, predicting the location's average service time $\hat{\overline{y}}_l$.

To obtain the average service time, we first calculate the service time per customer by normalizing the service time by the number of customers for each delivery event at the location, and then adopt the average value of all delivery events. In this way, high dynamics of service time due to varying number of customers can be mitigated, and the inherent attributes of the location can be better captured.

Since the output contains both quantities and categories, we train the embedding module using a hybrid loss function $\mathcal{L}_{pre}$ as shown in Equation 4:

$$\mathcal{L}_{pre} = (\mathbf{x}_{quant} - \hat{\mathbf{x}}_{quant})^2 + \beta \sum_{k=1}^{K} (-x_{poi}^k \log \hat{p}_{poi}^k) + \alpha(\overline{y}_l - \hat{\overline{y}}_l)^2 \tag{4}$$

where K is the number of POI types, and α, β are two hyperparameters to balance three kinds of loss.

3.4 Location Similarity-Aware Support Set Completion

Location similarity-aware support set completion aims to complete the support set of a learning task with scarce learning samples using that of similar learning tasks based on location embeddings learnt in the previous stage.

Main Idea. Though the meta-learning is suitable for learning from limited learning samples, the extremely scarce or empty samples are still a problem for a learning task to adapt well. The key insight we use to address this problem is that similar locations are likely to have similar delivery patterns. Therefore, if we can complete the support set of a sample-insufficient learning task with samples associated with similar learning tasks, the prediction performance of the sample-insufficient learning task should be improved.

Implementation. Figure 7(b) depicts the process of the location similarity-aware support set completion. Since each learning task corresponds to the deliveries to a certain

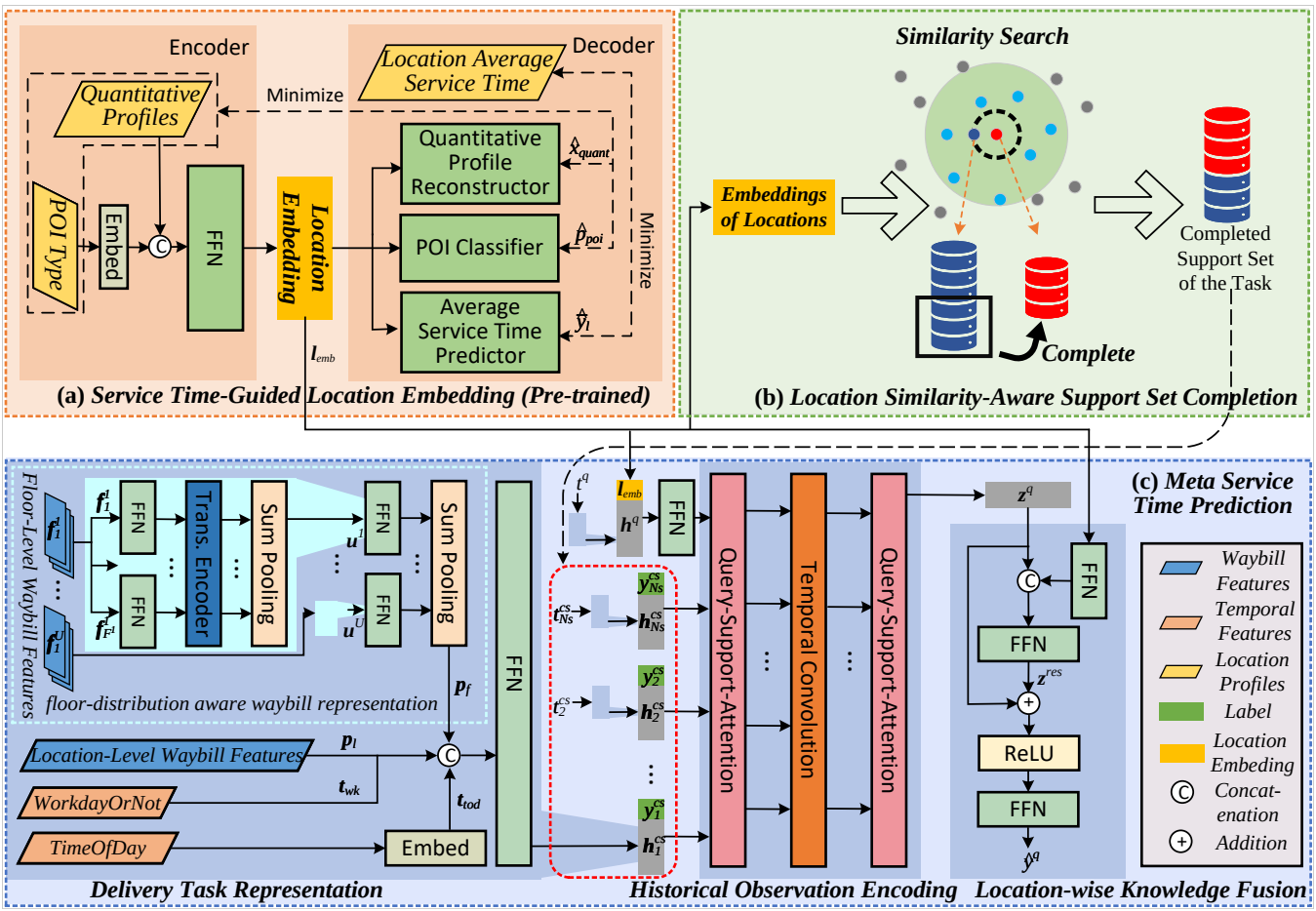


Fig. 7. The architecture of MetaSTP$^+$.

delivery location, we first project all learning tasks into the embedding space based on the previously learnt location embeddings. Then for each learning task, we check whether the size of its support set is smaller than a preset threshold, i.e., Num_{thd}. If its size is smaller than Num_{thd}, the support set completion process would be conducted. For a learning task whose support set needs to be completed, e.g., $\mathcal{T}_i$, we find a learning task that is within Dis_{thd} and is the closest to $\mathcal{T}_i$, e.g., $\mathcal{T}_j$ in the embedding space (according to the Euclidean distance in the embedding space). Finally, we choose a random subset of the support set $\mathcal{D}_j^s$ of $\mathcal{T}_j$ to complete the support set $\mathcal{D}_i^s$ of $\mathcal{T}_i^s$ to make the size of the completed support set reach Num_{thd}. We denote the completed support set of $\mathcal{T}_i$ as $\mathcal{D}_i^{cs}$.

3.5 Meta Service Time Prediction

Meta service time prediction aims to predict the service time for each delivery task leveraging the completed support set and the learnt location embedding, which consists of three modules as depicted in Figure 7(c).

- **Delivery Task Representation**, which extracts and embeds each delivery task's floor-level waybill features along with other features to obtain the fine-grained hidden representation of each delivery task;
- **Historical Observation Encoding**, which generates a vector that encodes the correlation among the query task, the location prior knowledge, and tasks with labels in the completed support set;
- **Location-wise Knowledge Fusion**, which further enhances the output vector with the location prior knowledge so that an ideal prediction can still be achieved even if the support set has no or very limited observations.

3.5.1 Delivery Task Representation

The delivery task representation module aims to encode each delivery task into an expressive representation to facilitate later observation correlation calculation and prediction.
Main Idea. This module aims to capture fine-grained waybill information as shown in the bottom left part of Figure 7. The module first groups waybills in the task by units and floors, and extracts waybill features for each floor involved. Then the set of floor-level features are jointly considered to obtain a floor distribution-aware waybill representation for each unit. Those representations from all units are further fused to obtain the floor-level waybill features. At last, it is combined with location-level waybill features as well as temporal features to further enrich the representation.

The floor distribution-aware waybill representation is inspired by three key insights from a delivery task:

1) The transition time at the same floor is much smaller than that among different floors (which involve waiting for elevators or walking upstairs) as shown in Figure 2(a).
2) Waybills distributed among all floors in a unit jointly determine its service time, as observed in Figure 2(b).
3) The service times of all units contribute to the overall time cost, but they are less affected by each other due to the unit by unit delivery as illustrated in Figure 2(c).

Implementation. First, we identify the unit and floor information from the address of each waybill based on regular expression (e.g., `Floor XX`, `Unit X`, `Building X`, ...). Waybills in the delivery tasks are then grouped.

Then, for each floor in each unit involved in the delivery task, we extract the following 5 floor-level features which potentially contribute to the service time by aggregating waybills at the floor: (1) the floor number; (2) the number of customers to deliver parcels to; (3) the total number of waybills; (4) the total weight of parcels; and (5) the total volume of parcels. Formally, we use $\mathbf{f}_i^j$ to denote floor-level features of the i^{th} floor of the j^{th} unit in the delivery task. Then, all floor-level features of a delivery task can be represented as a set $\{\mathbf{F}^j\}_{j=1}^{U}$, where $\mathbf{F}^j = \{\mathbf{f}_i^j\}_{i=1}^{F^j}$, F^j is the total number of floors in the j^{th} unit, and U is the total number of units in the task.

Next, we extract floor distribution-aware waybill representation $\mathbf{p}_f$ for the task based on $\{\mathbf{F}^j\}_{j=1}^{U}$, which is a two-stage fusion (floor-stage and unit-stage). In the floor stage fusion, the correlation between floors in the same unit are captured with a transformer encoder [17] after applying a feedforward network on floor-level features in $\mathbf{F}^j$. Then through a sum pooling we obtain the floor distribution-aware waybill representation of each unit, denoted as $\mathbf{u}^j$:

$$\mathbf{u}^j = \texttt{SumPool}(\texttt{TransEnc}(\texttt{FFN}(\mathbf{F}^j))) \tag{5}$$

where `FFN` contains a fully connected (FC) layer.

To obtain $\mathbf{p}_f$, we leverages a feedforward network to transform $\{\mathbf{u}^j\}_{j=1}^{U}$ into a hidden space, then the unit-level sum pooling is applied:

$$\mathbf{p}_f = \texttt{SumPool}(\texttt{FFN}(\{\mathbf{u}^j\}_{j=1}^{U})) \tag{6}$$

where `FFN` contains one FC layer and a ReLU activation.

Together with $\mathbf{p}_f$, we extract 6 location-level waybill features, which describe waybills in the task in a macro view: (1) the total number of customers to deliver, (2) the total number of waybills, (3) the total weight and (4) volume of parcels, (5) the number of units to deliver, and (6) the total number of floors involved in the addresses of waybills. The obtained feature vector is denoted as $\mathbf{p}_l$.

We further extract two temporal features from the planned delivery time slot. (1) $\mathbf{t}_{wk}$, whether today is workday or weekend. (2) $\mathbf{t}_{tod}$, embedding of the time slot within a day, where [8:00-23:00] is equally divided into 5 bins.

We concatenate $\mathbf{p}_f$, $\mathbf{p}_l$, $\mathbf{t}_{wk}$ and $\mathbf{t}_{tod}$, and then send them to a feedforward network to obtain the delivery task representation $\mathbf{h}$.

$$\mathbf{h} = \texttt{FFN}([\mathbf{p}_f; \mathbf{p}_l; \mathbf{t}_{wk}; \mathbf{t}_{tod}]) \tag{7}$$

where `FFN` contains 2 FC layers with ReLU activation, and ; means concatenation.

The representation from the support set $\{\mathbf{h}_i^s\}_{i=1}^{N_s}$ and that of the query delivery task $\mathbf{h}^q$ are all sent to the next module.

3.5.2 Historical Observation Encoding

After the previous delivery task representation, all delivery tasks in the completed support set $\{t_i^{cs}\}_{i=1}^{N_{cs}}$ are transformed into dense representations $\{\mathbf{h}_i^{cs}\}_{i=1}^{N_{cs}}$, and the query task t^q is transformed into $\mathbf{h}^q$. For each $\mathbf{h}_i^{cs}$, we concatenate it with its label y_i^{cs}, and obtain a dense representation $\mathbf{o}^{cs}$ of the support observation, i.e., $\mathbf{o}_i^{cs} = [\mathbf{h}_i^{cs}; y_i^{cs}]$. The dense representation of the support set is denoted as $\mathcal{D}^{cs'} = \{\mathbf{o}_i^{cs}\}_{i=1}^{N_{cs}}$.

Taking $\mathcal{D}^{cs'}$ and $\mathbf{h}^q$ as inputs, the historical observation encoding aims to learn the correlation among them, and generate an embedding vector $\mathbf{z}_q$ (as shown in the bottom middle part of Figure 7). $\mathbf{z}_q$ semantically is a high-dimensional representation of the prediction after "seeing" the completed support set at the location.

A straightforward idea to accomplish this is to use a classical model-based meta-learning approach [18], which proposes to interleave self-attention [17] with temporal convolution [19] to encode past experiences. It enjoys the benefit of accepting infinite large past experiences (from self-attention) and having a high-bandwidth to direct access a batch of past experiences (from temporal convolution). However, this approach has two limitations. Firstly, it only cares about the correlation between the query sample and observations in the completed support set, ignoring the difference of sample importance due to different location prior knowledge. Secondly, it equally treats samples in the query set and the support set, while we care more about making an accurate prediction based on the query sample.

Main Idea. Our idea is to fuse the location prior knowledge before the sample correlation calculation and replace the original self-attention layer with a *query-support attention* layer, so that the importance of each sample in the completed support set is related to both the query sample and the location prior knowledge, and the importance of the query sample can be emphasized.

Implementation. To fuse the location prior knowledge, we first concatenate the query task representation $\mathbf{h}^q$ with the location embedding $\mathbf{l}_{emb}$ and then create a prior knowledge-enhanced query representation $\mathbf{o}^{keq}$ by transforming it into the same size with $\mathbf{o}^{sc}$ using a feedforward network:

$$\mathbf{o}^{keq} = \texttt{FFN}([\mathbf{h}^q; \mathbf{l}_{emb}]) \tag{8}$$

where `FFN` contains one FC layer.

After that, $\{\mathbf{o}_1^{cs}, \mathbf{o}_2^{cs}, ..., \mathbf{o}_{N_{cs}}^{cs}, \mathbf{o}^{keq}\}$ are sent into an especially designed query-support attention layer, followed by a temporal convolution layer [19]. Then, another query-support attention layer is applied to make sure the past experiences are fully utilized.

In order to make the meta predictor emphasize more on the query sample, the query-support attention layer only leverages the attention weights between the query sample and support samples to make the representation transformation, rather than different attention weights in the self-attention mechanism. It first calculates the importance of each historical observation in the completed support set with respect to the prior knowledge-enhanced query representation $\mathbf{o}^{keq}$, i.e., attention weights $\boldsymbol{\alpha}$, based on the dot-product attention (Equation 9). Then, for the prior knowledge-enhanced query representation, the output is the concatenation of $\mathbf{o}^{keq}$ with the weighted sum of value representations of the support set (Equation 10), and for each support representation in the completed support set, the output is the concatenation of $\mathbf{o}^{cs}$ with its weighted value representation (Equation 11).

$$\boldsymbol{\alpha} = \texttt{softmax}(\frac{\mathbf{q}_{\mathbf{o}^{keq}} \mathbf{K}_{\mathbf{O}^{cs}}^{\top}}{\sqrt{d_{\mathbf{K}}}}) \tag{9}$$

$$\mathbf{o}^{keq'} = [\mathbf{o}^{keq}; \boldsymbol{\alpha}\mathbf{V}_{\mathbf{O}^{cs}}] \tag{10}$$

$$\mathbf{o}^{cs'} = [\mathbf{o}^{cs}; \alpha\mathbf{v}_{\mathbf{o}^{cs}}] \tag{11}$$

where $\mathbf{O}^{cs}$ represents the matrix $[\mathbf{o}_1^{cs}, \mathbf{o}_2^{cs}, ..., \mathbf{o}_{N_{cs}}^{cs}]$, $\mathbf{q}_{\mathbf{o}^{keq}} = \texttt{FC}(\mathbf{o}^{keq})$, $\mathbf{K}_{\mathbf{O}^s} = \texttt{FC}(\mathbf{O}^{cs})$ and $\mathbf{V}_{\mathbf{O}^s} = \texttt{FC}(\mathbf{O}^{cs})$.

We take the output from the position of query sample of the last query-support attention layer, i.e., $\mathbf{z}^q$, as the output of the historical observation encoding module, which contains the knowledge about how to make the service time prediction for the query delivery task with location prior knowledge-aware historical observations encoded.

3.5.3 Location-wise Knowledge Fusion

The location-wise knowledge fusion module takes historical observation encoded representation $\mathbf{z}^q$ from the previous module, further enhances it with location-wise prior knowledge, and gives the final service time prediction (as shown in the bottom right part of Figure 7).

Main Idea. Though the historical observation encoding module considers the location prior knowledge, the major role of it is to determine the importance of different observations. Here, we emphasize the importance of the prior knowledge again by the late fusion to explicitly consider the benefits of it for the service time prediction.

Implementation.

A straightforward approach is then to concatenate $\mathbf{l}_{emb}$ with $\mathbf{z}^q$ to make the final prediction. However, this strategy performs badly according to our experimental results. We guess the reason is that the output of historical observation encoding already contains rich information for the time prediction, while $\mathbf{l}_{emb}$ is a little bit noisy.

Inspired by ResNet [20], we propose to learn a residual information based on $\mathbf{z}^q$ and $\mathbf{l}_{emb}$ that is able to refine $\mathbf{z}^q$, and ultimately make the prediction more accurate, which is formally defined as follows:

$$\hat{y}^q = \texttt{FFN}(\texttt{ReLU}(\mathbf{z}^q + \texttt{FFN}([\mathbf{z}^q; \texttt{FFN}(\mathbf{l}_{emb})]))) \tag{12}$$

where FFN to calculate the residual contains 2 FC layers, and the first FC layer is followed by a ReLU activation following [20], FFN to obtain the output and to transform the original location prior knowledge contains one FC layer.

3.6 Optimization

To optimize the meta service time predictor, we need to choose the loss function $\mathcal{L}$ for each learning task (delivery location). Here, we employ MSE, which is widely used for the regression problem:

$$\mathcal{L}(\hat{y}^q, y^q) = (y^q - \hat{y}^q)^2 \tag{13}$$

Then, the meta service time predictor is trained end to end by minimizing meta loss $\mathscr{L}$ (Equation 2) with the above learning task loss $\mathcal{L}$ (Equation 13). The overall training procedure of MetaSTP$^+$ is given in Algorithm 1.

Algorithm 1 MetaSTP$^+$ Training Algorithm.

Input: Delivery task datasets $\mathcal{D}$; location profiles $\mathscr{F}_s$; location embedding encoder $g_{\boldsymbol{\omega}}$, location embedding decoder $h_{\boldsymbol{\phi}}$; MetaSTP model $f_{\boldsymbol{\theta}}$; query set rate r; minimum size of query set N_q^{min}, the support set completion threshold Num_{thd}; the embedding similarity threshold Dis_{thd}.
Output: The optimized parameters $\boldsymbol{\omega}$ of location embedding encoder, $\boldsymbol{\theta}$ of MetaSTP$^+$.

group $\mathcal{D}$ by locations to construct location task datasets $\mathscr{D}$;
for location delivery task dataset $\mathcal{D}_i \in \mathscr{D}$ **do**
  calculate N_q based on Equation 3 with $\mathcal{D}_i$, r and N_q^{min};
  randomly split $\mathcal{D}_i$ into $\mathcal{D}_i^s$, $\mathcal{D}_i^q$ according to N_q;
// Stage 1: Pre-train the location embedding module.
calculate the average service time $\overline{y}_l$ for each location based on $\mathscr{D}$;
randomly initialize $\boldsymbol{\omega}$ and $\boldsymbol{\phi}$;
repeat
  randomly select a batch of profiles $\mathscr{F}_{s\_batch}$ from $\mathscr{F}_s$;
  $\mathcal{L}_{pre\_batch} \leftarrow 0$;
  for $\mathbf{F}_s \in \mathscr{F}_{s\_batch}$ **do**
    $\mathcal{L}_{pre\_batch} \leftarrow \mathcal{L}_{pre\_batch} + \mathcal{L}_{pre}(h_{\boldsymbol{\phi}}(g_{\boldsymbol{\omega}}(\mathbf{F}_s)), \overline{y}_l, \mathbf{F}_s)$
  update $\boldsymbol{\omega}$ and $\boldsymbol{\phi}$ by minimizing $\mathcal{L}_{pre\_batch}$;
until *stopping criteria is met*
// Stage 2: Complete the support set.
infer the location embedding set $\mathcal{E}$ by $g_{\boldsymbol{\omega}}(\mathscr{F}_s)$;
for location delivery task dataset $\mathcal{D}_i \in \mathscr{D}$ **do**
  $\mathcal{D}_i^{cs} \leftarrow \mathcal{D}_i^s$;
  if $|\mathcal{D}_i^s| \leq Num_{thd}$ **then**
    $j \leftarrow get\_most\_similar\_location(\mathbf{l}_{emb}^i, \mathcal{E}, Dis_{thd})$;
    if j is not None **then**
      $\mathcal{D}_i^{cs} \leftarrow \mathcal{D}_i^s \cup random\_subset(\mathcal{D}_j^s, Num_{thd} - |\mathcal{D}_i^s|)$;
// Stage 3: Meta-train the meta service time predictor.
randomly initialize $\boldsymbol{\theta}$;
repeat
  randomly select a batch of learning tasks $\mathscr{D}_b$ from $\mathscr{D}$;
  $\mathscr{L} \leftarrow 0$;
  for location delivery task dataset $\mathcal{D}_l \in \mathscr{D}_b$ **do**
    $\mathscr{L} \leftarrow \mathscr{L} + \frac{1}{N_q} \sum_{(t_i^q, y_i^q) \in \mathcal{D}_l^q} \mathcal{L}(f_{\boldsymbol{\theta}}(t_i^q, \mathcal{D}_l^{cs}, g_{\boldsymbol{\omega}}(\mathbf{F}_s)), y_i^q)$;
  update $\boldsymbol{\theta}$, $\boldsymbol{\omega}$ by minimizing $\mathscr{L}$;
until *stopping criteria is met*
return $\boldsymbol{\theta}$, $\boldsymbol{\omega}$;

4 Experiments

4.1 Datasets

Our datasets consist of historical delivery tasks and spatial external knowledge, which are introduced as follows.

- **Historical Delivery Tasks.** We use two real world datasets from JD Logistics for evaluation, which are collected in the downtown area (DowBJ) and suburban area (SubBJ) of Beijing (splitted by the 3rd Ring) over a period of 20 months (from Jan. 1st, 2018 to Sept. 1st, 2019). The raw data consist of couriers' trajectories and waybills. We first use our previous work [9], [10] to infer the delivery location of each waybill, then group waybills in each delivery trip into delivery tasks. The pending delivery time slot of each task is set according to the start time of each delivery trip, since STP is usually called before departure. Finally, we match each delivery task to a stay point (detected from couriers' trajectories) according to the accurately annotated delivery time or the spatial closeness [9]. The duration of the stay point is treated as the service time of the corresponding task. After the previous data pre-processing steps, we obtain a database consisting

TABLE 1
Statistics of Datasets ("/" is train/val/test separator).

Datasets	DowBJ	SubBJ
#Delivery Tasks	53,979/5,978/6,138	22,403/4,630/5,520
#Delivery Locations	1,166/628/591	1,520/865/1,018
#New Loc. in Val/Test	13/16	266/424
Avg. Service Time (s)	418	395

of historical delivery tasks as well as their corresponding service times. We use the data from the first 16 months as training set, the data from the following 2 months as validation set, and use the last 2 months for testing. The details of each dataset are summarized in Table 1.

- **Spatial External Knowledge.** For each delivery location, we obtain its POI type via reverse Geocoding [2], which contains 18 POI types, and the built year and the second-hand house price for Residence are crawled from the Web [3], while for other types of POI, we use the mean to fill missing values.

We provide the distribution of some important aspects of both datasets as follows.

Number of Observations Distribution. Figure 8(a) shows the distribution of the number of observations of a delivery location in both datasets. As can be seen, the observations are distributed highly skewed in the urban space. For 80% locations in DowBJ, there are less than 69 observations for training. The case is even worse in SubBJ, which only have less than 19 observations under the same criteria.

Number of Customers Distribution. Figure 8(b) shows the distribution of the number of customers involved in a delivery task. As observed, both datasets have similar distribution, for around 40%-50% delivery tasks, couriers have to deliver parcels for more than one customer at a location, which introduces many uncertainties for STP.

Number of Floors & Units Distribution. Figure 8(c) shows the distribution of the number of floors of a delivery task in both datasets, which also show similar distribution. For around 40% delivery tasks, couriers need to go to different floors to complete the delivery task at those locations. In addition, there are 10% delivery tasks in both datasets, in which couriers need to go to multiple building units at a deliver location.

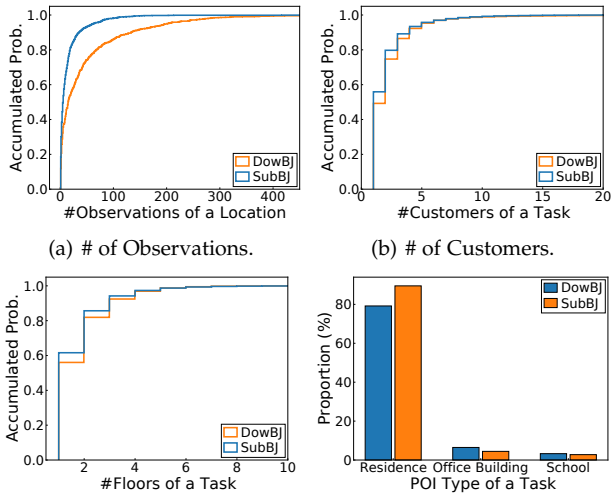
POI Types Distribution. Figure 8(d) shows the distribution of POI type of a delivery task in both datasets. The top 3 POI types are the same in both datasets: Residence, Office Building, and School. The deliveries for Residence take up for around 80%. It can also be noticed that the portion of deliveries for Office Building is a bit higher in DowBJ than SubBJ, which is consistent with our common sense.

4.2 Experimental Settings

Baselines. We compare MetaSTP$^+$ with following baselines.

- HA, which gives the historical average service time.
- HLA, giving the location-specific historical average value.
- HCA, assuming the time to be proportional to the number of customers with a factor estimated from training data.
- GBRT [21], which trains a gradient boosting regression tree based on historical observations to make prediction.

2. https://lbs.qq.com/
3. https://www.fang.com/



(a) # of Observations. (b) # of Customers.

(c) # of Floors. (d) POI Types.

Fig. 8. Data distributions.

- MLP [22], which trains a 3-layer MLP to make prediction.
- DeepFM [23], which captures both low- and high-order feature interactions in delivery tasks by combining factorization machines (FM) and deep neural networks.
- Song et al. [11], which is the earliest work focusing on addressing the service time prediction problem. It trains a KNN regressor based on aggregated features from waybills in a delivery task [24].
- GPD [25], a generative pre-training framework for spatio-temporal few-shot learning with urban knowledge transfer.
- PWT (SOTA) [3], which is a deep service time prediction model considering the number of floors the courier needs to climb based on aggregated waybills.

For all machine learning baselines, features from waybills only contain location-level ones, which are concatenated with others as input, since it is impractical to pad the floor-level waybill features to the maximum length given long-tailed involved floors in samples.

Variants. We also compare MetaSTP$^+$ with following variants to show the effectiveness of its components.

- MetaSTP$^+$-nMeta, which removes the meta-learning component and is trained in the way like traditional machine learning methods.
- MetaSTP$^+$-nLP, which removes all designs about incorporating the location prior knowledge. That is, the entire method only preserves the delivery task representation module and the classic meta-learning method is used [18].
- MetaSTP$^+$-nPre, which removes the pre-trained location embedding stage, i.e., the support set completion and prior knowledge incorporation are implemented directly based on raw features of locations.
- MetaSTP$^+$-nTS, which removes the sub-decoder of Average Service Time Predictor, so that the pre-trained location embedding stage is not guided by average service time.
- MetaSTP$^+$-nC, which removes the location similarity-aware support set completion stage.
- MetaSTP$^+$-nSeq, which removes the floor distribution-aware waybill representation $\mathbf{p}_f$ from MetaSTP.
- MetaSTP$^+$-nPHE, which replaces the newly designed

prior knowledge-aware historical observation encoding module with the classic module used in the model-based meta-learning [18].

- MetaSTP⁺-nQSA, which replaces the newly designed PHE module as MetaSTP⁺-nPHE does. However, unlike MetaSTP⁺-nPHE, it integrates location prior knowledge before modeling the correlations among samples. Like MetaSTP⁺-nPHE, it does not emphasize the importance of query sample.
- MetaSTP⁺-OOA, which replaces our newly designed attention layer in PHE module by replacing α and V in Eq. (11) with the attention weights and values corresponding to other historical observations in the completed support set. Its attention also can emphasize the importance of the query sample. And it also fuses location prior knowledge before modeling the correlation of samples.
- MetaSTP⁺-nLKF, which drops the location-wise knowledge fusion module, and predicts the time based on the output from historical observation encoding via a FC.
- MetaSTP⁺-nRes, which drops the residual connection, and directly concatenates $\mathbf{l}$ with the output from the historical observation encoding to make the prediction.
- MetaSTP [8], which is the prior version of this work.

Evaluation Metrics. We leverage three commonly used metrics for regression problem, i.e., MAE, RMSE and MAPE, to evaluate the performance of different methods. MAE $= \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$, where N is the total number of delivery tasks in the test set. MAE characterizes the average prediction error with respect to the ground-truth over all test samples. RMSE $= \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$, which is more sensitive to samples with large prediction errors. MAPE $= \frac{100\%}{N}\sum_{i=1}^{N}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$, which measures the average relative errors of the prediction and the ground-truths.

Training Details & Hyperparameters. Our method as well as baselines are completely implemented in Python 3.8 using PyTorch 1.11 on a docker with 16 Cores@2.2GHz, 64GB memory and Ubuntu 22.04 Linux. We have released our code to help understand more implementation details [4]. To setup the meta-training dataset, for each learning task, $r = 0.2$ and $N_q^{min} = 1$. (1) In the Service Time-Guided Location Embedding stage, the hidden size of the pre-trained location encoder, the quantitative profile reconstructor, the POI classifier and the average service time predictor are $[16, 8, 4, 2]$, $[16, 8, 4, 4]$, $[8, 32, 16, 18]$ and $[8, 4, 2, 1]$, respectively. β and α are set to 1.2, 8.0 and 1.2, 4.0 for DowBJ and SubBJ, respectively. (2) In the Location Similarity-Aware Support Set Completion stage, the completion threshold Num_{thd} and the similarity threshold Dis_{thd} are set to 8, 0.2 and 28, 0.2 for DowBJ and SubBJ, respectively. (3) In the Meta Service Time Prediction stage, we leverage Adam with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ to meta-train with a learning rate $3e$-3. We sample 4 learning task in each iteration, and the whole meta-training dataset is iterated over 8 times. The hidden size of FC before the transformer, in the transformer encoder, in the representation output as well as in the self-attention are all set to 8. FC to obtain $\mathbf{u}$ contains 16 neurons. WorkdayOrNot is embedded to $\mathbb{R}^3$. The temporal convolution is stacked by 4 dilated 1D convolutions with 16 filters. POI Type is embedded to $\mathbb{R}^2$. To learn $\mathbf{z}_{res}$, the hidden size is 2. The hyperparameters of baselines are also selected based on the best validating performance.

4. https://github.com/YangQY2000/metastp-plus

TABLE 2
Overall Evaluation.

Methods	DowBJ			SubBJ		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE
HA	253.9	368.1	126.0	229.6	320.3	112.0
HLA	204.2	295.0	88.2	205.8	282.9	93.8
HCA	174.0	269.3	68.7	157.7	235.5	64.4
MLP	161.4	257.7	65.9	150.0	259.5	61.5
GBRT	156.4	239.5	67.0	152.0	241.4	56.9
DeepFM [23]	156.2	239.5	63.2	157.1	250.1	56.2
Song et al. [11]	154.4	237.4	63.5	153.5	223.0	60.7
GPD [25]	156.2	237.2	66.5	150.8	239.6	65.2
PWT (SOTA) [3]	153.6	233.9	66.1	149.6	236.6	57.0
MetaSTP⁺-nMeta	149.8	232.5	59.4	145.8	246.2	56.3
MetaSTP⁺-nRes	144.3	224.4	54.6	141.9	282.7	51.8
MetaSTP⁺-nLP	144.1	229.4	54.8	141.2	241.6	53.8
MetaSTP⁺-nSeq	141.3	215.9	55.8	139.6	271.3	54.9
MetaSTP [8]	139.7	226.9	48.9	138.5	216.0	50.0
MetaSTP⁺-nC	139.1	219.0	50.3	138.8	211.4	49.6
MetaSTP⁺-nPre	138.8	215.7	50.2	139.7	210.9	52.5
MetaSTP⁺-nPHE	137.9	216.1	48.9	138.5	211.8	48.8
MetaSTP⁺-nQSA	137.7	215.9	48.7	138.3	211.1	49.7
MetaSTP⁺-nLKF	137.7	217.6	48.6	138.2	207.9	51.4
MetaSTP⁺-OOA	137.2	216.6	48.4	138.0	210.5	49.2
MetaSTP⁺-nTS	137.0	215.7	47.9	138.1	209.3	49.9
MetaSTP⁺-nW	197.0	292.2	77.5	191.2	288.4	71.4
MetaSTP⁺-nSE	139.4	217.4	49.0	145.1	213.7	55.8
MetaSTP⁺-nT	138.8	215.0	52.1	137.7	209.3	**47.0**
MetaSTP⁺ (Ours)	**136.4**	**214.3**	**47.6**	**137.1**	**206.3**	47.7

4.3 Evaluation

Overall Performance. The overall performance of MetaSTP⁺ compared with baselines over all three metrics is shown in Table 2. As can be observed, directly predicting the service time based on historical average (HA) leads to huge errors, indicating that it is far from enough to empirically estimate the service time without considering the information of a certain delivery task. HLA is better than HA, which shows the uniqueness of different locations. HCA is better than HLA, which shows the number of customers surely is a shared strong signal affects the time among different locations. Traditional machine learning methods (MLP, GBRT, and KNN) show superior performance than aforementioned empirical ones, since they are able to model various factors by aggregating waybills in the delivery task. Nevertheless, the coarse-grained location-level features is not representative enough. GPD, considers the heterogeneity of different tasks. However, it requires informative spatial and temporal prompts for each task, in which the temporal prompts are derived from historical observations of a location, but we have many locations with extremely scarce historical observations. The state-of-the-art (SOTA) method, PWT, considers the floor information, leading to better results. However, the delivery task representation is still not fine-grained enough and the universal model is not able to fit locations with few observations well. Those limitations leave us the room for improvements. Our method, i.e., MetaSTP⁺, not only encodes the delivery task into a finer-grained level, but also leverages meta-learning and the prior knowledge to tackle the problem of skewed observations among locations. MetaSTP⁺ consistently outperforms baselines over three metrics on two datasets. Its MAE is 136.4s on DowBJ and 137.1s on SubBJ, which outperforms the best baseline by 11.2% and 8.4%, respectively.

Ablation Study. The ablation study is also conducted in Table 2 to validate the effectiveness of different components of MetaSTP⁺. After removing the meta-learning strategy (MetaSTP⁺-nMeta), a significant performance drop is witnessed, which shows the advantages of using meta-learning to tackle the skewed observation issues. The prior knowledge about the location is also vital for the service time prediction. As can be observed, without incorporating the prior knowledge, the prediction performance is severely degraded, which shows the necessity of introducing the prior knowledge-enhanced spatial meta-learning (compared with MetaSTP⁺-nLP). If we ignore the encoding for floor-level waybill features (MetaSTP⁺-nSeq), the performance degradation is also obvious, indicating that the floor distribution-aware waybill representation learning did provide finer-grained information for service time prediction. When removing the pre-trained location embedding (MetaSTP⁺-nPre), there is also a performance drop, which shows the raw location profiles are not appropriate to be directly used as the prior knowledge. Comparing with MetaSTP⁺-nPHE, we see that emphasizing the importance of query sample and leveraging prior knowledge before modeling the correlation of samples is helpful to the performance. Comparing MetaSTP⁺-nQSA with MetaSTP⁺-nPHE, it shows that fusing location prior knowledge before modeling the correlation of samples is beneficial. MetaSTP⁺-OOA (its attention is of a different design from *query-support attention* layer, but also to emphasize the importance of query sample) performs better than MetaSTP⁺-nQSA, which shows that emphasizing the importance of query sample is beneficial to the time prediction. But it performs worse than MetaSTP⁺, which shows the necessity of a subtly designed attention layer for the service time prediction problem.

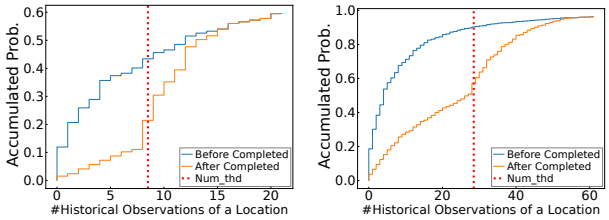
MetaSTP⁺-nTS shows that introducing the average service time prediction task in the pre-training stage can further boost the prediction performance. The support set completion is also an important stage. If the prediction is made without it, as MetaSTP⁺-nC shown, the performance would degenerate, which shows the necessity of the completion and proves its effectiveness to tackle the problem of skewed observations. Comparing with MetaSTP⁺-nLKF, we see the location embedding is helpful to be fused into the time prediction. Comparing MetaSTP⁺-nRes with MetaSTP⁺-nLKF, we find if the prior knowledge is not subtly fused, the performance might drop significantly. Moreover, MetaSTP⁺-nRes even causes a bit more performance drop than MetaSTP⁺-nLP, which indicates that the location prior knowledge is useful but noisy, and the residual connection can help reducing the effect of noise while keeping the benefit of the prior knowledge. Lastly, MetaSTP⁺ outperforms our prior work MetaSTP by 5.6% and 4.5% in RMSE in DowBJ and SubBJ, respectively due to the newly introduced pre-trained location embedding, location similarity-aware support set completion and the newly presented meta service time predictor.

Performance On Locations Triggering the Completion. As is shown in Table 3, we evaluate MetaSTP⁺, MetaSTP, and the state-of-the-art baseline PWT on locations with scarce observations. We define a *data-scarce location* as one whose support set needs to be completed. The results indicate that meta-learning approaches improve performance on scarce locations compared to PWT. We notice that the performance metrics on data-scarce location are better than the overall ones. We guess that this is due to higher sample diversity in locations with abundant samples, which leads to greater errors. However, focusing on data-scarce location enhances the ability to handle newly seen locations, which is still of high values.

TABLE 3
Performance On Locations Triggering the Completion.

Methods	DowBJ			SubBJ		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE
PWT(SOTA)	147.0	212.1	67.2	149.8	222.2	60.3
MetaSTP	128.5	192.9	52.9	134.7	208.7	55.4
MetaSTP⁺ (Ours)	**122.2**	**183.1**	**49.4**	**128.4**	**188.5**	**52.5**

Changes in the Number of Support Set Samples After Completion. As shown in Figure 9, before the completion, the proportion of data-scarce location with fewer than Num_{thd} historical observations is 0.43 and 0.90 on the two datasets, respectively. It proves the wide existence of extremely data-scarce locations. But after the completion, the proportions decrease significantly. On average, a location with scarce observations gains 6.4 and 15.8 additional samples during the completion operation on DowBJ and SubBJ, respectively. Although our completion operation aims to complete a location's support set up to the size of Num_{thd} (with its most similar location's samples), a proportion of locations still have fewer than Num_{thd} samples after the completion, it is reasonable because some data-scarce location identifies another data-scarce location as its most similar one, limiting the number of samples it can gain through completion.



(a) DowBJ. (b) SubBJ.

Fig. 9. The distribution of the number of support samples before and after the support set completion.

Importance of Features. MetaSTP⁺-nW, MetaSTP⁺-nSE and MetaSTP⁺-nT are MetaSTP⁺ without features from waybills, spatial external knowledge and temporal information dropped. As expected, the features from waybills play the most important role for the service time prediction. The features from spatial and temporal domain also contribute to the prediction, which shows the complexity of the service time prediction. However, though the temporal features are effective, they are more important in DowBJ, (especially the MAPE of MetaSTP⁺-nT in SubBJ is even a bit lower). We guess this phenomenon may attribute the less population in suburban areas, where the crowdness in a building does not change much across different time periods and the noise of temporal features overwhelmed their effect.

Different Weights of Losses in Pre-training. Recalling that the loss function of the location embedding pre-training has three parts, which are controlled by the weights for the

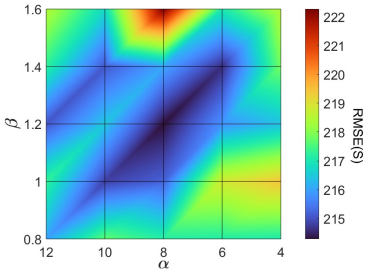
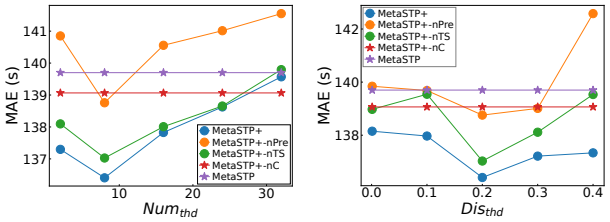


Fig. 10. Different weights of pre-training losses.

loss of the average service time prediction (α) and the POI classification (β). Figure 10 shows RMSE of MetaSTP$^+$ with respect to α and β on DowBJ. It can be observed, when α=8 and β=1.2, RMSE reaches the minimum. Changing α or β makes the prediction error larger, which suggests a trade-off between the average service time prediction task and the location profile reconstruction tasks. This experiment indicates the importance of the average service time prediction task, which proves the effectiveness of enhancing the location embedding with semantics of average service time.

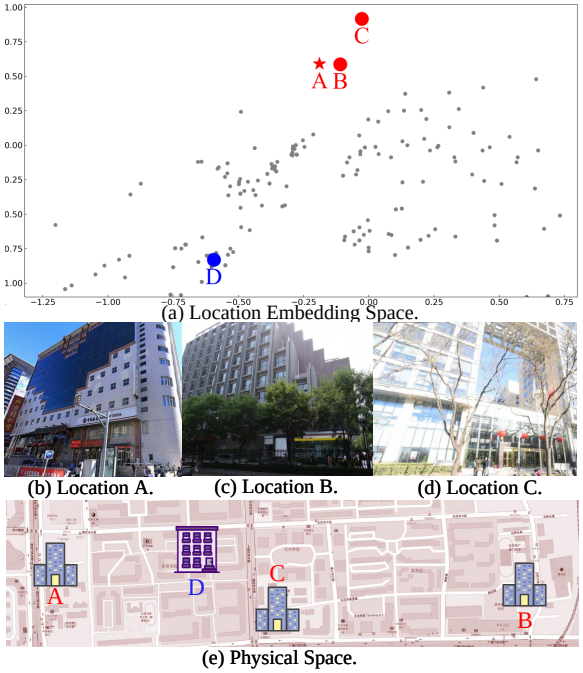
Different Completion Thresholds. Num_{thd} is one of the key hyper-parameters in the support set completion, which is a threshold for the size of a support set to determine whether the support set of a learning task should be completed. Figure 11(a) shows the change of MAE of MetaSTP$^+$ and 2 variants, which derives task similarities in different ways (MetaSTP$^+$-nPre and $^+$-nTS) with the increase of Num_{thd} on DowBJ. And two methods without the support set completion (MetaSTP$^+$-nC and MetaSTP) are also added for reference. When Num_{thd} is 8, all methods requiring the support set completion, perform their best, which indicates Num_{thd} is a trade-off hyper-parameter. When Num_{thd} is too small, only support sets with a size close to 0 are completed, and the skewed observation problem is not fully resolved. When Num_{thd} is too large, the performance would also decline, because too many samples from other locations might dominate the samples of the current location, which makes the model difficult to capture the uniqueness of the current location. MetaSTP$^+$ consistently outperforms MetaSTP$^+$-nPre and $^+$-nTS in different Num_{thd}, indicating the effectiveness of the service time-guided location embedding. The comparison with MetaSTP$^+$-nC and MetaSTP also shows that Num_{thd} should be carefully set.



(a) Different Num_{thd}. (b) Different Dis_{thd}.

Fig. 11. Different hyperparameters for the support set completion.

Different Task Similarity Thresholds. Dis_{thd} is the other key hyper-parameter in the support set completion, which judges the similarity of different learning tasks in the location embedding space. Figure 11(b) shows the change of MAE of MetaSTP$^+$ as well as the same set of variants with the increase of Dis_{thd} on DowBJ. As is shown, when Dis_{thd} is 0.2, all methods requiring the support set completion, perform their best, which indicates Dis_{thd} is also a trade-off hyper-parameter. When Dis_{thd} is too small, the criteria to judge whether a location is similar is too strict. Thus, for many learning tasks with limited observations, the support set cannot be completed due to task dissimilarity. On the other hand, when Dis_{thd} is too large, the completion process may judge many locations with large embedding distances as similar locations of the current one. It makes the support set of the current location be completed with many samples from locations with quite different delivery patterns, and thus leads to great prediction errors. Similar to Num_{thd}, the service time-guided location embedding makes MetaSTP$^+$ outperform other methods to derive the task similarities in different Dis_{thd}, and Dis_{thd} should also be carefully set.



(a) Location Embedding Space.

(b) Location A. (c) Location B. (d) Location C.

(e) Physical Space.

Fig. 12. Case study of location embedding.

Case Study of Location Embedding. To show the effectiveness of the location embedding, we give a case study to illustrate whether the location embedding works as we expected, i.e., locations with similar delivery patterns are close in the embedding space. We plot the embeddings of different locations in Figure 12(a). As shown in the figure, we find Location A is close to Location B, has a moderate distance to Location C, and far away from Location D in the embedding space. Apart from Location D is a residential building, Location A, B, and C are office buildings as the street views shown in Figure 12(b), (c) and (d). But Locations A and B have some shops on the first floor, while Location C does not, implying that the location embedding module did map locations with similar delivery patterns closer in the embedding space. We further show those locations in the physical space in Figure 12(e). We find that semantic-closer locations are not necessarily spatial-closer, showing the necessity of learning the latent embeddings of locations.

5 Deployment

An intelligent waybill assignment system based on MetaSTP$^+$ is used internally in JD Logistics to balance the

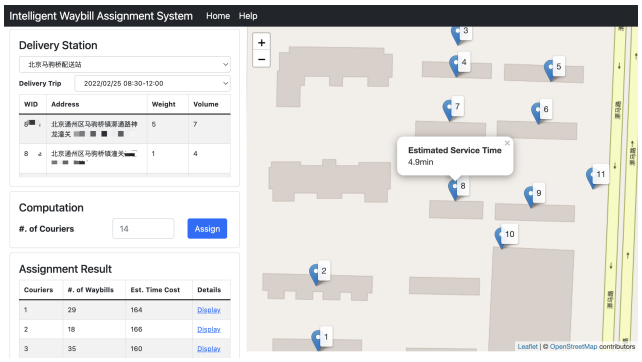


Fig. 13. System interface.

working hours of couriers.

The interface of the system is shown in Figure 13. The station master can select one of delivery trips of today, and the details of the batch of waybills pending to be delivered would be listed in the table. After the number of couriers to conduct the delivery for the batch of waybills is entered, the batch of waybills would be assigned to couriers by solving the distance/time-constrained capacitated vehicle routing problem (DCVRP) [26]. The locations in DCVRP are derived from the waybills, the travel time between locations is estimated from couriers' historical transitions, and the service time at each location is predicted by MetaSTP$^+$.

The system is deployed on Cloud Virtual Machine in JD Cloud with the trained MetaSTP$^+$ model and a commercial MIP solver. The historical observations, which acts as the support set, are obtained from Redis. The waybills allocated to the station are accessed from an internal Web API. When deployed online, MetaSTP$^+$ can predict the service time for 250 tasks/s per thread. The multi-thread processing is further implemented to increase its throughput. For a courier, the error of delivery trip time estimation is reduced by 15 minutes, which greatly balances the working hours of couriers at the delivery station.

6 Related Work

Stay Time Prediction. The stay time prediction focuses on modeling the time cost of a user staying at a specific POI. Chen et al. [27] use GBRT based on various spatio-temporal features extracted from historical stay points at that location, Liu et al. [28] leverage MLP considering spatio-temporal features as well as context logs in smartphones, and Gidófalv et al. [29] propose to use Markov models based on individual's previous stay sequence. STP differs from the stay time prediction in that the underlying semantics of stay are specific and different from general ones, i.e., they are driven by the delivery tasks but not others, e.g., working, eating. In stay time prediction, what a moving object is doing at a location usually is unknown, and can only be inferred from sequential patterns or smartphone logs, while we explicitly know the stay is caused by delivery, and the delivery task should be leveraged. The earliest work focusing on STP is [11], which predicts the service time using a KNN regressor. However, [11] extracts features by aggregating waybills, ignoring the fine-grained floor information. A recent work [3] realized the importance of the number of floors to deliver, however the features from waybills are also aggregated. Moreover, both of them did not tackle the issue of skewed observations, which makes performance of the prediction model deteriorated.

Estimated Time of Arrival (ETA). ETA aims to estimate the time that a vehicle/person is expected to arrive at its destination [30], [31]. A special case for ETA is that there are multiple intermediate destinations, e.g., parcel deliveries [4], [32], [33], [34], [35], [36], where the service time at each location is a part of its time cost. Existing multi-destination ETA usually implicitly considers the service time, which is modeled into the time cost between consecutive locations. However, individually predicting the service time not only can be used in it, but also benefit other downstream applications, e.g., workload balancing via service zone partition [3] and route planning with strict time-windows [1], [2].

Meta-Learning Applications in Spatio-temporal Data Mining. Meta-learning has been adopted in the field of spatio-temporal data mining when data is limited. For example, Yao et al. [37] make spatio-temporal prediction in cities with limited data, and Qin et al. [38] predict the purchase volume in different regions and day types. Xu et al. [39] adaptively predict personalized spatial trajectory. Recently, Yuan et al. [25] propose a generative framework for spatio-temporal few-shot learning with urban knowledge transfer, which generates a dedicated network for each task. Jiang et al. [40] propose Spatio-Temporal Meta-Graph Learning, and implement this idea into Meta-Graph Convolutional Recurrent Network (MegaCRN) for traffic forecasting. In addition, in the map search engine, Fang et al. [7] estimate the en route travel time, Fan et al. [41] auto-complete POI and Chen et al. [42] predict the next POI to search. Different from these studies, we are the first work to leverage meta-learning to improve the predictability of service time in logistics. A representation layer is specially designed to embed the delivery task. Besides, in our scenario, the support set could be empty, which is not the case for the aforementioned works. We design a prior knowledge fusion module to enhance the performance under this case.

Delivery Data Mining. With the digitization progress of the logistics industry, there are emerging studies focusing on delivery data mining. In addition to pick-up/arrival time estimation [4], [32], [33], [34], [35], [36] and STP [3], [11], there are many other studies of delivery data mining, which are reviewed as follows. Based on waybill data, Ding et al. [43] infer the delivery scope of each merchant, and Wen et al. [44], [45], [46] predict the pick-up order of parcels. Guo et al. [3] balance the workload of couriers via partitioning the delivery zone based on the estimated service time. With couriers' trajectories, Dahiya et al. [47] find the regions of interest, Srivastava et al. [48] improve the quality of Geocoding, [9], [10], [49] infer the delivery location, and Jiang et al. [50] detect fake locations registered by the merchants. Leveraging couriers' encounter data, Ding et al. [51] estimate the relative location of couriers indoors.

7 Conclusion

In this paper, we study service time prediction (STP), which is fundamental for intelligent logistics, and propose MetaSTP$^+$ to solve it. MetaSTP$^+$ treats STP at each location as a learning task to tackle the location heterogeneity, and

leverages a fine-grained representation layer to encode the complex delivery circumstances. A three-stage spatial meta-learning framework is proposed to tackle the skewed observations which injects the location prior knowledge before, during and after the meta-learning process. MetaSTP$^+$ outperforms baselines by at least 11.2% and 8.4% on two real-world datasets, and is used internally in JD Logistics.

Though in this work, we focus on STP, the idea of comprehensive prior knowledge injection by support set completion, query-support attention, and late knowledge fusion essentially can be easily transferred to other problems. It has potential to empower the modeling of other learning problems with a set of related tasks, which have prior knowledge. Furthermore, in the future, we envision a more universal framework to tackle the spatial activity modeling problem.

ACKNOWLEDGMENTS

We thank Shengnan Wu and Yiheng Chen from JD Logistics, and Zheyi Pan, Zhipeng Ma from JD Technology for the early discussion of this work. This research is supported by the National Key R&D Program of China (2023YFC3321505), National Natural Science Foundation of China (62306033, 42371480). This paper is an extended version of an earlier paper published at the 28th SIGKDD Conference (KDD 2022) [8].

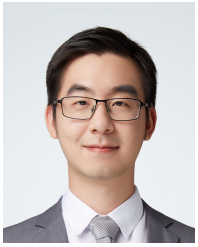
REFERENCES

[1] M. Desrochers, J. Desrosiers, and M. Solomon, "A new optimization algorithm for the vehicle routing problem with time windows," Operations research, vol. 40, no. 2, pp. 342–354, 1992.
[2] W. P. Nanry and J. Wesley Barnes, "Solving the pickup and delivery problem with time windows using reactive tabu search," Transportation Research Part B: Methodological, vol. 34, no. 2, pp. 107–121, 2000.
[3] B. Guo, S. Wang, H. Wang, Y. Liu, F. Kong, D. Zhang, and T. He, "Towards equitable assignment: Data-driven delivery zone partition at last-mile logistics," in KDD, 2023.
[4] F. Wu and L. Wu, "Deepeta: A spatial-temporal sequential neural network model for estimating time of arrival in package delivery system," in AAAI, vol. 33, no. 01, 2019, pp. 774–781.
[5] H. Zhang, H. Wu, W. Sun, and B. Zheng, "Deeptravel: A neural network based travel time estimation model with auxiliary supervision," in IJCAI, vol. 19, 2018, pp. 3655–3661.
[6] Z. Wang, K. Fu, and J. Ye, "Learning to estimate the travel time," in KDD, 2018, pp. 858–866.
[7] X. Fang, J. Huang, F. Wang, L. Liu, Y. Sun, and H. Wang, "Ssml: Self-supervised meta-learner for en route travel time estimation at baidu maps," in KDD, 2021, pp. 2840–2848.
[8] 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 KDD, 2022.
[9] 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 KDD, 2020, pp. 2813–2821.
[10] S. Ruan, C. Long, X. Yang, T. He, R. Li, J. Bao, Y. Chen, S. Wu, J. Cui, and Y. Zheng, "Discovering actual delivery locations from mis-annotated couriers' trajectories," in ICDE. IEEE, 2022.
[11] J. Song, R. Wen, C. Xu, and J. W. E. Tay, "Service time prediction for last-yard delivery," in Big Data. IEEE, 2019, pp. 3933–3938.
[12] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in ICML. PMLR, 2017, pp. 1126–1135.
[13] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, "Meta-learning in neural networks: A survey," TPAMI, 2021.
[14] Y. Liu, J. Ding, Y. Fu, and Y. Li, "Urbankg: An urban knowledge graph system," ACM Transactions on Intelligent Systems and Technology, vol. 14, no. 4, pp. 1–25, 2023.
[15] Y. Liu, X. Zhang, J. Ding, Y. Xi, and Y. Li, "Knowledge-infused contrastive learning for urban imagery-based socioeconomic prediction," in WWW, 2023, pp. 4150–4160.
[16] Y. Yan, H. Wen, S. Zhong, W. Chen, H. Chen, Q. Wen, R. Zimmermann, and Y. Liang, "Urbanclip: Learning text-enhanced urban region profiling with contrastive language-image pretraining from the web," in WWW, 2024, pp. 4006–4017.
[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in NIPS, 2017, pp. 5998–6008.
[18] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel, "A simple neural attentive meta-learner," in ICLR, 2018.
[19] A. Van Den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," SSW, 2016.
[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in CVPR, 2016, pp. 770–778.
[21] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," Annals of statistics, pp. 1189–1232, 2001.
[22] I. Goodfellow, Y. Bengio, and A. Courville, Deep learning. MIT, 2016.
[23] H. Guo, R. TANG, Y. Ye, Z. Li, and X. He, "Deepfm: A factorization-machine based neural network for ctr prediction," in IJCAI-17, 2017.
[24] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," The American Statistician, vol. 46, no. 3, pp. 175–185, 1992.
[25] Y. Yuan, C. Shao, J. Ding, D. Jin, and Y. Li, "Spatio-temporal few-shot learning via diffusive neural network generation," in ICLR, 2024.
[26] A. G. Kek, R. L. Cheu, and Q. Meng, "Distance-constrained capacitated vehicle routing problems with flexible assignment of start and end depots," Mathematical and Computer Modelling, vol. 47, no. 1-2, pp. 140–152, 2008.
[27] J. Chen, Z. Xiao, D. Wang, W. Long, J. Bai, and V. Havyarimana, "Stay time prediction for individual stay behavior," IEEE Access, vol. 7, pp. 130 085–130 100, 2019.
[28] S. Liu, H. Cao, L. Li, and M. Zhou, "Predicting stay time of mobile users with contextual information," T-ASE, vol. 10, no. 4, pp. 1026–1036, 2013.
[29] G. Gidófalvi and F. Dong, "When and where next: Individual mobility prediction," in Proceedings of the First ACM SIGSPATIAL International Workshop on Mobile Geographic Information Systems, 2012, pp. 57–64.
[30] J. Han, H. Liu, S. Liu, X. Chen, N. Tan, H. Chai, and H. Xiong, "ieta: A robust and scalable incremental learning framework for time-of-arrival estimation," in KDD, 2023, pp. 4100–4111.
[31] H. Liu, W. Jiang, S. Liu, and X. Chen, "Uncertainty-aware probabilistic travel time prediction for on-demand ride-hailing at didi," in KDD, 2023, pp. 4516–4526.
[32] C. Gao, F. Zhang, G. Wu, Q. Hu, Q. Ru, J. Hao, R. He, and Z. Sun, "A deep learning method for route and time prediction in food delivery service," in KDD, 2021, pp. 2879–2889.
[33] L. Zhang, X. Zhou, Z. Zeng, Y. Cao, Y. Xu, M. Wang, X. Wu, Y. Liu, L. Cui, and Z. Shen, "Delivery time prediction using large-scale graph structure learning based on quantile regression," in ICDE. IEEE, 2023, pp. 3403–3416.
[34] Y. Qiang, H. Wen, L. Wu, X. Mao, F. Wu, H. Wan, and H. Hu, "Modeling intra- and inter-community information for route and time prediction in last-mile delivery," in ICDE, 2023, pp. 3106–3112.
[35] H. Wen, Y. Lin, F. Wu, H. Wan, Z. Sun, T. Cai, H. Liu, S. Guo, J. Zheng, C. Song, and L. Wu, "Enough waiting for the couriers: Learning to estimate package pick-up arrival time from couriers' spatial-temporal behaviors," TIST, 2023.
[36] T. Cai, H. Wan, F. Wu, H. Wen, S. Guo, L. Wu, H. Hu, and Y. Lin, "M2g4rtp: A multi-level and multi-task graph model for instant-logistics route and time joint prediction," in ICDE, 2023, pp. 3296–3308.
[37] H. Yao, Y. Liu, Y. Wei, X. Tang, and Z. Li, "Learning from multiple cities: A meta-learning approach for spatial-temporal prediction," in WWW, 2019, pp. 2181–2191.
[38] H. Qin, S. Ke, X. Yang, H. Xu, X. Zhan, and Y. Zheng, "Robust spatio-temporal purchase prediction via deep meta learning," in AAAI, vol. 35, no. 5, 2021, pp. 4312–4319.

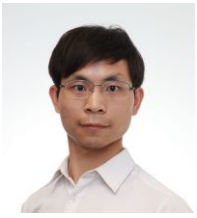
[39] Y. Xu, J. Xu, J. Zhao, K. Zheng, A. Liu, L. Zhao, and X. Zhou, "Metaptp: An adaptive meta-optimized model for personalized spatial trajectory prediction," in KDD, 2022, p. 2151–2159.
[40] R. Jiang, Z. Wang, J. Yong, P. Jeph, Q. Chen, Y. Kobayashi, X. Song, S. Fukushima, and T. Suzumura, "Spatio-temporal meta-graph learning for traffic forecasting," in AAAI, vol. 37, no. 7, 2023, pp. 8078–8086.
[41] M. Fan, Y. Sun, J. Huang, H. Wang, and Y. Li, "Meta-learned spatial-temporal poi auto-completion for the search engine at baidu maps," in KDD, 2021, pp. 2822–2830.
[42] Y. Chen, X. Wang, M. Fan, J. Huang, S. Yang, and W. Zhu, "Curriculum meta-learning for next poi recommendation," in KDD, 2021, pp. 2692–2702.
[43] X. Ding, R. Zhang, Z. Mao, K. Xing, F. Du, X. Liu, G. Wei, F. Yin, R. He, and Z. Sun, "Delivery scope: A new way of restaurant retrieval for on-demand food delivery service," in KDD, 2020, pp. 3026–3034.
[44] H. Wen, Y. Lin, F. Wu, H. Wan, S. Guo, L. Wu, C. Song, and Y. Xu, "Package pick-up route prediction via modeling couriers' spatial-temporal behaviors," in ICDE. IEEE, 2021, pp. 2141–2146.
[45] H. Wen, Y. Lin, H. Wan, S. Guo, F. Wu, L. Wu, C. Song, and Y. Xu, "Deeproute+: Modeling couriers' spatial-temporal behaviors and decision preferences for package pick-up route prediction," TIST, vol. 13, no. 2, pp. 1–23, 2022.
[46] H. Wen, Y. Lin, X. Mao, F. Wu, Y. Zhao, H. Wang, J. Zheng, L. Wu, H. Hu, and H. Wan, "Graph2route: A dynamic spatial-temporal graph neural network for pick-up and delivery route prediction," in KDD, 2022, p. 4143–4152.
[47] M. Dahiya, D. Samatia, and K. Rustogi, "Learning locality maps from noisy geospatial labels," in SAC, 2020, pp. 601–608.
[48] V. Srivastava, P. Tejaswin, L. Dhakad, M. Kumar, and A. Dani, "A geocoding framework powered by delivery data," in SIGSPATIAL, 2020, pp. 568–577.
[49] Y. Song, J. Li, L. Chen, S. Chen, R. He, and Z. Sun, "A semantic segmentation based poi coordinates generating framework for on-demand food delivery service," in SIGSPATIAL, 2021, pp. 379–388.
[50] D. Jiang, Y. Ding, H. Zhang, Y. Liu, T. He, Y. Yang, and D. Zhang, "Alwaes: an automatic outdoor location-aware correction system for online delivery platforms," IMWUT, vol. 5, no. 3, pp. 1–24, 2021.
[51] Y. Ding, D. Jiang, Y. Yang, Y. Liu, T. He, and D. Zhang, "P2-loc: A person-2-person indoor localization system in on-demand delivery," IMWUT, 2022.

Shuliang Wang received the Ph.D. degree from Wuhan University and Hong Kong Polytechnic University, in 2002. He is currently a Professor of Beijing Institute of Technology, Executive Dean of the E-Government Institute. His research interests include spatial data mining, and big data intelligence. He was awarded the first prize of National Science and Technology Award for Science and Technology Progress, National Excellent Doctoral Dissertation, and so on.

Qianyu Yang is a PhD student at the School of Computer Science, Beijing Institute of Technology. His research interests include spatio-temporal data mining and deep learning.



Sijie Ruan is currently an Assistant Professor at the School of Computer Science and Technology, Beijing Institute of Technology. He received his Ph.D. and B.E. degrees from Xidian University in 2022 and 2017, respectively. His research interests include spatio-temporal data mining and urban computing. Before joining BIT, he was a visiting Ph.D. student at Nanyang Technological University, and research interns at Microsoft Research Asia and JD Technology.



Cheng Long (S'11-M'15-SM'22) is currently an Associate Professor at the College of Computing and Data Science, Nanyang Technological University. From 2016 to 2018, he worked as a lecturer at Queen's University Belfast, UK. He received his PhD degree from the Hong Kong University of Science and Technology, Hong Kong, in 2015, and his BEng degree from South China University of Technology, China, in 2010. His research interests are broadly in data management and data mining.

Ye Yuan received his B.E., M.E., and Ph.D. degrees in computer science from Northeastern University, in 2004, 2007, and 2011, respectively. He is now a professor with the School of Computer Science and Technology, Beijing Institute of Technology, China. His research interests include graph databases, probabilistic databases, and social network analysis.

Qi Li is currently a lecture of Beijing Forestry University. He received his Ph.D. and M.E. degrees from Beijing Institute of Technology in 2021 and 2017, respectively. He was a post-doctoral researcher in BIT from 2021 to 2024. His research interests include data mining and machine learning.

Ziqiang Yuan is currently a PhD student at the School of Computer Science and Technology, Beijing Institute of Technology. He received his M.E. and B.E. degrees from Beijing Institute of Technology in 2019 and 2017. His research interests include QA systems and probability graphs.

Jie Bao got his Ph.D degree in Computer Science from University of Minnesota at Twin Cities in 2014. He worked as a researcher in Urban Computing Group at Microsoft Research Asia from 2014 to 2017. He currently leads the data management department at JD Intelligent Cities Business Unit, JD Technology. His research interests include: Spatio-temporal Data Management/Mining, Urban Computing, and Location-based Services.

Yu Zheng is the Vice President of JD.COM and president of JD Intelligent Cities Research. Before Joining JD.COM, he was a senior research manager at Microsoft Research. He is also a chair professor at Shanghai Jiao Tong University. He was the Editor-in-Chief of ACM Transactions on Intelligent Systems and Technology and has served as the program co-chair of ICDE 2014 and CIKM 2017. He is a keynote speaker of AAAI 2019, KDD 2019 Plenary Keynote Panel and IJCAI 2019 Industrial Days. He received SIGKDD Test-of-Time Award twice (in 2023 and 2024). He was named one of the Top Innovators under 35 by MIT Technology Review (TR35), an ACM Distinguished Scientist and an IEEE Fellow, for his contributions to spatio-temporal data mining and urban computing.