Graph2Route: A Dynamic Spatial-Temporal Graph Neural Network for Pick-up and Delivery Route Prediction

Haomin Wen
Beijing Jiaotong University
Cainiao Network
wenhaomin@bjtu.edu.cn

Youfang Lin
Xiaowei Mao
Beijing Jiaotong University
Beijing Key Laboratory of Traffic Data Analysis and Mining

Fan Wu
Cainiao Network
wf118503@cainiao.com

Yiji Zhao
Haochen Wang
Beijing Jiaotong University
yijizhao@bjtu.edu.cn
wanghaochen@bjtu.edu.cn

Jianbin Zheng
Lixia Wu
Haoyuan Hu
Cainiao Network

Huaiyu Wan*
Beijing Jiaotong University
Beijing Key Laboratory of Traffic Data Analysis and Mining
hywan@bjtu.edu.cn

ABSTRACT

Pick-up and delivery (P&D) services such as food delivery have achieved explosive growth in recent years by providing customers with daily-life convenience. Though many service providers have invested considerably in routing tools, more and more practitioners realize that significant deviations exist between workers' actual routes and planned ones. So it is not wise to feed "optimal routes" as workers' actual service routes into downstream tasks (e.g., arrival-time prediction and order dispatching), whose performances count on the accuracy of route prediction, i.e., to predict the future service route of a worker's unfinished tasks. Therefore, to meet the rising calling for route prediction models that can capture workers' future routing behaviors, in this paper, we formulate the Pick-up and Delivery Route Prediction task (PDRP task for short) from the graph perspective for the first time, then propose a dynamic spatial-temporal graph-based model, named *Graph2Route*. Unlike previous sequence-based models, our model leverages the underlying graph structure and features into the encoding and decoding process. Moreover, the dynamic graph-based nature can spontaneously describe the evolving relationship between different problem instances. As a result, abundant decision context information and various spatial-temporal information of node/edge can be fully utilized in Graph2Route to improve the prediction performance. Offline experiments over two real-world industry-scale datasets under different P&D services (i.e., food delivery and package pick-up) and online A/B test demonstrate the superiority of our proposed model.

*Corresponding author
Emails: {yflin, maoxiaowei}@bjtu.edu.cn
{jabin.zjb, wallace.wulx, haoyuan.huhy}@cainiao.com

KDD '22, August 14–18, 2022, Washington, DC, USA

ACM ISBN 978-1-4503-9385-0/22/08…$15.00
https://doi.org/10.1145/3534678.3539084

CCS CONCEPTS

• **Information systems** → **Data mining**; • **Applied computing** → **Forecasting**.

KEYWORDS

pick-up and delivery service; route prediction, dynamic graph

1 INTRODUCTION

In recent years, many pick-up and delivery services (e.g., food delivery and logistics) are playing an increasingly important role in rapid urbanization and daily-life convenience. Though many services providers have invested considerably in routing tools, some practitioners realize that significant deviations exist between the workers' actual routes and the planned ones [3, 10, 17]. According to a research project conducted in the U.S. and Mexico, using the deliveries data over one year for a large soft drinks company, it was found that three out of four deliveries did not follow the planned routes [10]. As a consequence, it is not wise to feed "optimal routes" as workers' actual service routes into downstream tasks, such as arrival-time prediction [2, 15, 18] and order dispatching [5, 9, 14], whose performances always count on the accuracy of route prediction (i.e., to predict the future service route of a worker's unfinished tasks, such as picked/undelivered meals in the food delivery service). For example, accurate route prediction can reduce the uncertainty in service arrival time estimation, thus enabling the platform to provide more punctual services and alleviate customers' waiting anxiety [18]. The above examples urge the development of P&D route prediction methods that can accurately capture the future routing behaviors of workers (rather than planning an optimal route, and note that the two goals are fundamentally different).

Table 1: Comparison between our model and related ones.

Method	Modeling method	Modeling decision context?	Problem
OSquare [20]	Sequenced-based	No	Pick-up then Delivery
DeepRoute [17]	Sequenced-based	No	Pick-up
FDNET [3]	Sequenced-based	No	Pick-up then Delivery
Our model	**Graph-based**	**Yes**	**Pick-up & Delivery**

The PDRP problem has not been well studied in the academic community. Three most related works are OSquare [20], DeepRoute

[17] and FDNET [3]. OSquare [20] is the pioneer to solve the route prediction problem in food delivery. It converts the route prediction into a next-location prediction problem and outputs the whole route step by step with a traditional machine learning model. FDNET [3] proposes a deep model equipped with an LSTM [4] encoder and a Pointer [13] decoder to predict the driver's route at once. In the logistics field, DeepRoute [17] utilizes a Transformer-like encoder and an attention-based recurrent decoder to predict courier's future pick-up routes. Although the above models have achieved promising results, they still have the following limitations (the comparison of those methods and ours is shown in Table 1):

1) As shown in Figure 1(left), previous works all treat unfinished tasks (i.e., a problem instance) as a sequence. Thus sequence-based encoder (e.g., BiLSTM) is utilized for encoding them. However, the sequence-based nature of the encoders limits their ability to fully encode the spatial-temporal correlations between different tasks. 2) As shown at time t_2 in Figure 1, previous sequence-based methods cannot avoid decoding extremely unreasonable routes, e.g., the next location D is the furthest location from the previous location A, which significantly deviates from the worker's actual routes. 3) Previous models make the prediction only based on the information at the request time, ignoring the connections between the problem instances at different time steps. In real-life scenarios, however, the problem instances of a certain worker actually evolve over time (e.g., change of task requirements or coming of new tasks), leading to natural connections between nearby problem instances. Failing to model the above connections limits the performance of previous models.

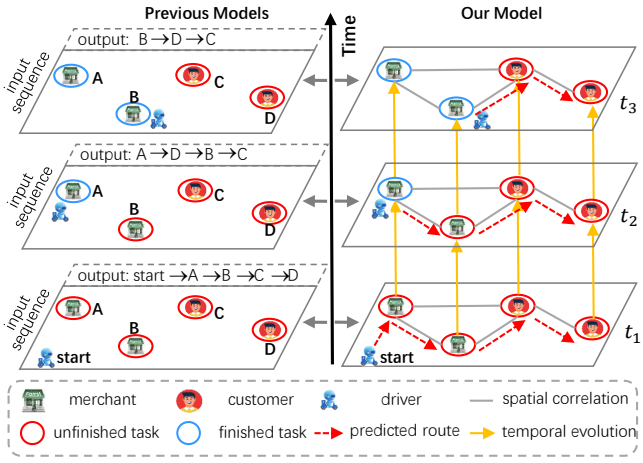

Figure 1: Illustration of Previous Models and Our Model. (Taking food delivery as background)

To cope with the above limits, we propose a novel dynamic spatial-temporal graph-based model, named *Graph2Route*, to predict a worker's future service route. As shown in Figure 1(right), all the tasks, including finished ones and unfinished ones, are distributed in the geographic space, and the spatial relationship between them can be naturally depicted as a graph. To this end, we model a problem instance from the graph perspective, so that the node/edge features and graph structure can be utilized to improve the prediction quality (for limitation 1). To address limitation 2, we incorporate the graph structure into the decoder, which can filter unreasonable solutions and reduce the search space. Moreover, to model the evolving relationship between consecutive problem instances, we develop a dynamic spatial-temporal graph neural network that can capture the decision context (i.e., decision-making environments a few steps before) to make more accurate predictions (for limitation 3). In summary, the main contributions of this paper are as follows:

- This is the first work to investigate the PDRP task from the dynamic graph perspective to the best of our knowledge. A dynamic spatial-temporal graph-based model is proposed to precisely capture the future routing behaviors of workers.
- A dynamic ST-Graph encoder is presented to encode the spatial-temporal correlations into the node and edge embeddings, where a recurrent neural network is designed to carry on the decision context for more accurate prediction.
- A graph-based personalized route decoder is designed to compute more reasonable and accurate future routes, while leveraging the graph structure and workers' personalized information.
- Extensive online A/B test and offline experiments conducted on two real-world datasets demonstrate that our method significantly outperforms other solutions.

2 PRELIMINARIES: GRAPH PERSPECTIVE

In this section, we provide a new perspective of formulating the PDRP task from the graph perspective.

Input ST-Graph. Different P&D services have different types of tasks, for example, there is only pick-up tasks in package pick-up service, while both pick-up and delivery task exist in the food delivery service. Without loss of generality, we represent each task as a node in the graph, and a problem instance of worker w at time t can be defined on a spatial-temporal graph (ST-graph) $\mathcal{G}_t^w = (\mathcal{V}_t, \mathcal{E}_t, \mathbf{X}_t, \mathbf{E}_t)$, where $\mathcal{V}_t = \{v_1, \ldots, v_n\}$, with each node corresponds to a task of the worker (e.g., driver or courier). $\mathcal{E}_t = \{(i,j) \mid v_i, v_j \in \mathcal{V}_t\}$ is the set of edges. $\mathbf{X}_t \in \mathbb{R}^{n \times d_v}$ and $\mathbf{E}_t \in \mathbb{R}^{n \times n \times d_e}$ are the node and edge features respectively, where d_v and d_e are the node feature dimension and edge feature dimension, respectively. Both of them contain the spatial-temporal information of different tasks and can be constructed by service-specific settings. For example, constructing nodes features according to the requirements of a task (e.g., location, promised service time), constructing edge features according to the spatial-temporal correlation between two tasks (e.g., distance, connectivity). Moreover, let $\mathbf{x} \in \mathbb{R}^{d_v}$ be the features of a task v. And x^{FT} is one of the features, denoting the finish time of the task. At a certain time t, there are two types of nodes in graph $\mathcal{G}_t^w$: i) finished nodes $\mathcal{V}_t^F = \{v \mid v \in \mathcal{V}_t, x^{FT} \leq t\}$, and ii) unfinished nodes $\mathcal{V}_t^U = \{v \mid v \in \mathcal{V}_t, x^{FT} = -1\}$.

Route Constraints. In reality, various route constraints exist in the P&D services, such as pick-up then delivery constraint (i.e., the delivery location of an order can only be visited after the pick-up location is visited [3]) and capacity constraint (i.e., the total weight of items carried by a worker can not exceed its capacity of load [12]). Route constraints of a problem can be represented by a rule set C, with each item corresponding to a specific route constraint.

Problem Definition. Given the input graph $\mathcal{G}_t^w$ of worker w at time t, we aim to learn a mapping function $\mathcal{F}_C$ to predict the worker's future service route $\hat{\boldsymbol{\pi}}$ of unfinished nodes which can satisfy the given route constraints C, formulated as:

$$\mathcal{F}_C(\mathcal{G}_t^w) = \pi_1, \pi_2 \cdots \pi_{|\mathcal{V}_t^U|}, \tag{1}$$

where π_i means that the i-th node in the route is node v_{π_i} and $|\cdot|$ denotes the cardinality of a set. Moreover, $\pi_i \in \{1, \cdots |\mathcal{V}_t|\}$ and $\pi_i \neq \pi_j$ if $i \neq j$. Figure 2 gives an illustration of the problem.

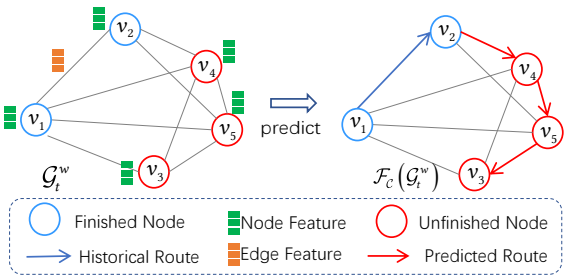

Figure 2: Problem Illustration. In this case, $\mathcal{V}_t^F = \{v_1, v_2\}$ and $\mathcal{V}_t^U = \{v_3, v_4, v_5\}$, the output of the model $\hat{\boldsymbol{\pi}} = [\pi_1, \pi_2, \pi_3]$ is $[4, 5, 3]$.

3 PROPOSED GRAPH2ROUTE MODEL

In this section, we introduce the dynamic spatial-temporal graph-based model, named Graph2Route, to solve the food delivery route prediction problem [3], for it is a typical P&D service and contains a common constraint (i.e., pick-up then delivery constraint).

A typical process of food delivery service includes three main steps: i) a customer places an order on the online platform; ii) the platform dispatches the order to an appropriate driver (worker), and iii) the driver finishes the order by executing two tasks within the promised time: pick-up task that picks up the food at its corresponding merchant, and delivery task that deliveries the food to the customer. Suppose that a driver w has a set of finished tasks F_t^w and unfinished tasks $U_t^w = \{task_p^{o_1}, task_d^{o_1}, \ldots, task_p^{o_m}\}$ at a certain time t, where $task_p^o$ and $task_d^o$ represents the pick-up and delivery task of order o, respectively. Based on the information of decision context, driver, and task requirements, we aim to produce an accurate prediction of the driver's delivery route, which is the permutation of all the unexecuted tasks in U_t^w.

Figure 3 illustrates the proposed Graph2Route, which is equipped with a dynamic ST-Graph encoder and graph-based personalized route decoder. The dynamic ST-Graph encoder takes the node and edges features at time t as input and converts them into embeddings, which can capture the spatial-temporal correlations and the evolution of decision context. And the decoder computes the predicted route based on the node embeddings recurrently, while taking the worker's personalized information and the underlying graph structure into account.

3.1 Input Spatial-Temporal Graph

Different from previous methods that encode the finished and unfinished tasks by separate modules [16], we represent them with a unified graph structure with no need to design two different encoding modules, thus the inter-relationship between them can be well captured by the graph-based methods. Specifically, in the food delivery service, an order is associated with two tasks for a driver (worker) once it is dispatched, i.e., pick-up task and delivery task. Each task is considered as a node in the graph. Given the finished task set and the unfinished task set of a worker at time t, we construct an input ST-graph $\mathcal{G}_t^w = (\mathcal{V}_t, \mathcal{E}_t, \mathbf{X}_t, \mathbf{E}_t)$, where node set $\mathcal{V}_t = F_t^w \cup U_t^w$ and edge set $\mathcal{E}_t = \{(i, j) \mid v_i, v_j \in \mathcal{V}_t \text{ and } a_{ij} \neq 0\}$. a_{ij} is the (i, j)-th entry of the adjacent matrix, defined as follows:

$$a_{ij} = \begin{cases} 1, \text{ if } i \text{ and } j \text{ are } k\text{−nearest spatial neighbors} \\ 1, \text{ if } i \text{ and } j \text{ are } k\text{−nearest temporal neighbors} \\ -1, \text{ if } i = j \\ 0, \text{ others.} \end{cases} \tag{2}$$

a_{ij} represents the proximity of i and j, which is utilized to guide the decoding process and can accelerate the learning process, as a node in the worker's actual route is usually connected to nodes in its proximity. The k-nearest spatial neighbors are defined according to the distance between nodes. And the temporal neighbors are defined by the time feature x_i^{PT} of each node, that is, if a node's promised time is close to another, they are called neighbors in the temporal dimension. Moreover, we use value -1 to represent self-connections and 0 for other conditions.

Node/Edge features. The node features and edges features are constructed as follows. For simplicity, we remove the subscript t of the local variables (such as $\mathbf{x}_i$ and $\mathbf{e}_{ij}$) in the following description.

Given a task (node) $i \in \mathcal{V}_t$, the node feature vector $\mathbf{x}_i$ contains spatial and temporal information related to the task and is formulated as follows:

$$\mathbf{x}_i = (x_i^{Co}, x_i^{AT}, x_i^{PT}, x_i^{FT}, t - x_i^{AT}, x_i^{PT} - t, x_i^{Dis}), \tag{3}$$

where x_i^{Co} is the geographic coordinates of the task, and x_i^{AT}, x_i^{PT} are the accepted time and promised arrival time, respectively. x_i^{Dis} is the distance of the task to the worker's current location.

Given an edge $(i, j) \in \mathcal{E}_t$, the edge feature is formulated as: $\mathbf{e}_{ij} = (d_{ij}, a_{ij})$. Where d_{ij} is the routing distance between the connected nodes, and a_{ij} is the proximity between the two nodes.

The input node and edge features are projected into d_h-dimension embeddings, $\overline{\mathbf{x}}_i$ and $\overline{\mathbf{e}}_{ij}$, by two linear transformations as follows:

$$\begin{aligned} \overline{\mathbf{x}}_i &= \sigma(\mathbf{W}_v * \mathbf{x}_i + \mathbf{b}_v), \\ \overline{\mathbf{e}}_{ij} &= \sigma(\mathbf{W}_e * \mathbf{e}_{ij} + \mathbf{b}_e), \end{aligned} \tag{4}$$

where σ is the ReLU activation function, and $\mathbf{W}_v \in \mathbb{R}^{d_v \times d_h}$, $\mathbf{b}_v \in \mathbb{R}^{d_h}$ and $\mathbf{W}_e \in \mathbb{R}^{d_e \times d_h}$, $\mathbf{b}_e \in \mathbb{R}^{d_h}$ are trainable parameters.

3.2 Dynamic ST-Graph Encoder

The dynamic ST-Graph encoder computes the node and edge embeddings at each time step, with two encoding modules: i) Spatial-Correlation encoding, a graph convolutional network to capture the spatial-temporal correlations between different nodes (e.g., spatial proximity). Unlike the standard GCN, which performs the information fusion process with only node features [8, 19], we also introduce edge features to jointly learn more comprehensive embeddings. ii) Temporal-Correlation encoding, an RNN is utilized to model the evolution of the decision context and carry on historical information by regulating the embeddings in consecutive steps.

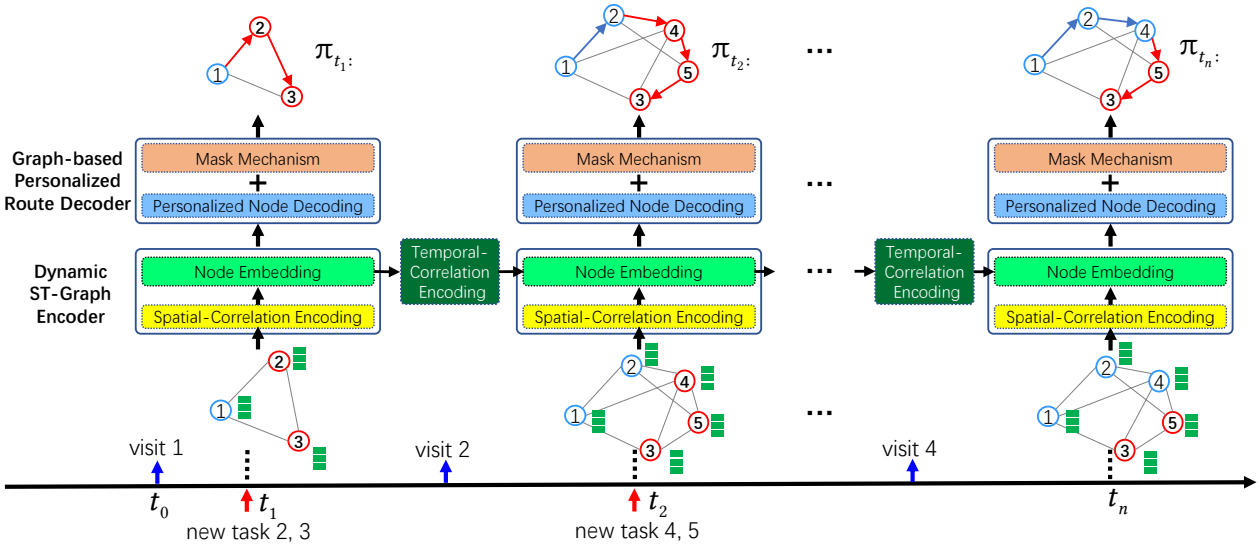

Figure 3: Architecture of Graph2Route.

Spatial-Correlation Encoding. We design a Spatial-Correlation Encoding block to model the spatial correlation. Given the initial d_h-dimensional node embedding and edge embedding, the GCN encoder layer computes the embeddings through L graph convolutional layers (GCN), with each layer updating the node and edge embeddings by leveraging their interactions.

Let $\mathbf{h}_i^l$ denote the embedding of node i at the l−th layer, and $\mathbf{z}_{ij}^l$ denote the edge embedding at the l−th layer. The initial embeddings come from the input layer, i.e., $\mathbf{h}_i^0 = \overline{\mathbf{x}}_i$ and $\mathbf{z}_{ij}^0 = \overline{\mathbf{e}}_{ij}$. Most GCNs perform information fusion by aggregating node embeddings from one-hop neighbors recursively. In this work, we jointly update the node and edge embeddings in the information updating process, which can be formulated as:

$$\begin{aligned} \mathbf{h}_i^{l+1} &= f(\mathbf{h}_i^l, \text{Agg}\{\mathbf{h}_j^l, \mathbf{z}_{ij}^l : j \in \mathcal{N}_i\}) \\ \mathbf{z}_{ij}^{l+1} &= g(\mathbf{z}_{ij}^l, \text{Agg}\{\mathbf{h}_i^l, \mathbf{h}_j^l\}), \end{aligned} \tag{5}$$

where $\mathcal{N}_i$ denotes the set of neighbors centered at node i, Agg$(\cdot)$ is the aggregation function, and the updating function f, g is composed by a non-linear transformation, defined as follows:

$$\mathbf{h}_i^{l+1} = \mathbf{h}_i^l + \sigma(\text{BN}(\mathbf{W}_1^l \mathbf{h}_i^l + \sum\nolimits_{j \in \mathcal{N}_i} \eta_{ij}^l \odot \mathbf{W}_2^l \mathbf{h}_j^l)) \tag{6}$$

$$\mathbf{z}_{ij}^{l+1} = \mathbf{z}_{ij}^l + \sigma(\text{BN}(\mathbf{W}_3^l \mathbf{z}_{ij}^l + \mathbf{W}_4^l \mathbf{h}_i^l + \mathbf{W}_5^l \mathbf{h}_j^l)), \tag{7}$$

where $\mathbf{W}_i^l \in \mathbb{R}^{d_h \times d_h}$ $(i = 1, \ldots, 5)$ are trainable parameters, σ is ReLU activation function. BN$(\cdot)$ represents batch normalization. And $\eta_{ij}^l = \sigma(\mathbf{W}_6^l \mathbf{z}_{ij}^l) / \sum_{j' \in \mathcal{N}_i} \sigma(\mathbf{W}_6^l \mathbf{z}_{ij'}^l)$, where $\mathbf{W}_6^l \in \mathbb{R}^{d_h \times 1}$ is a trainable parameter. After L graph convolution layers, we get the output of Spatial-Correlation encoding: $\mathbf{H}_t$ and $\mathbf{Z}_t$.

Temporal-Correlation Encoding. We design a Temporal-Correlation Encoding block to model the temporal correlation. According to a survey in the Cainiao logistics platform, couriers tend to give more coherent and consistent route arrangements. That is, the route arrangement at a certain time t always correlates with those before t, especially with nearby ones. For example, assume that no new-dispatched packages between t_1 and t_2, then the routing decisions at the two steps should be the same or at least share some similarity.

To model the temporal connections between consecutive steps, we introduce the historical node information to update the current node embeddings. Specifically, at each time step, the encoder takes the graph $\mathcal{G}_t^w$ and node embeddings $\mathbf{H}_{t-1}$ from the previous step as input, and output the node embedding matrix $\mathbf{H}_t$ at time t. The heart of the dynamic graph encoder is the update of the node embeddings $\mathbf{H}_t$ at time t based on the current, as well as the historical information (decision context). This requirement can be naturally fulfilled through the recurrent architecture, by treating the embeddings as the hidden state of the dynamical system. We use a gated recurrent unit (GRU) for node embeddings, which is formulated as:

$$\mathbf{H}_t = \text{GRU}(\mathcal{G}_t^w, \mathbf{H}_{t-1}). \tag{8}$$

The above process can be implemented by a standard GRU, with one extension, i.e., extending the inputs and hidden state from vectors to matrices (because the hidden state is now the node embedding matrix). One straightforward method is placing the row vectors of the matrix side by side to form a long vector. In other words, one uses a single GRU to process the long vector unfolded by the node embedding matrix.

Overall, the encoder first computes the node and edge embeddings by leveraging their interactions, then updates the node embeddings efficiently based on the prior ones. The dominant advantage of such a way is fully utilizing spatial-temporal features and considering the decision context.

3.3 Graph-based Personalized Route Decoder

The route decoder forecasts future route using the output embeddings from the encoder, with: i) a mask mechanism for meeting route constraints and ii) a personalized node decoding module for unfinished nodes selection at each step.

Mask Mechanism. It is developed to mask infeasible nodes at decoding step j. We use $\mathcal{N}_i$ to denote node i's neighbors, $\mathcal{R}_j = \{v_{\pi_1}, \ldots, v_{\pi_{j-1}}\} \subseteq \mathcal{V}$ to denote already outputted nodes before decoding step j, and v_d^o, v_p^o to denote the delivery, pick-up node of an order o, respectively. The mask mechanism includes the following four rules:

1) Mask the already finished nodes $\mathcal{V}_t^F$ before the prediction time t. Since there is no need to predict the order of them. 2) Mask the already outputted nodes before decoding step j, i.e., $\mathcal{R}_j$. 3) Mask the delivery node of an order whose pick-up node is not visited, i.e., $\mathcal{V}_j^d = \{v_d^o \mid v_p^o \notin \mathcal{V}_t^F \text{ and } v_p^o \notin \mathcal{R}_j\}$. Because in the food delivery service, an order's delivery location can only be visited after its corresponding pick-up location is visited. 4) Mask the nodes which are not the neighbors of latest outputted node $v_{\pi_{j-1}}$ according to the input graph, formulated as $\{\mathcal{V} - \mathcal{N}_{v_{\pi_{j-1}}}\}$. Because in the real world, workers always tend to choose a nearby location to visit next. By leveraging the graph structure in the mask mechanism, some unlikely solutions will be cut off, significantly reducing the search space and improving the model performance. Overall, the masked node set $\mathcal{V}_{mask}^j$ at decoding step j is: $\mathcal{V}_{mask}^j = \mathcal{V}_t^F \cup \mathcal{R}_j \cup \mathcal{V}_j^d \cup \{\mathcal{V} - \mathcal{N}_{v_{\pi_{j-1}}}\}$.

During the decoding process, the mask should also be updated at each step, that is, if the last outputted node is a pick-up node, then we should remove the mask of its corresponding delivery node, so that the delivery node can be added into the candidate at the next decoding step. The implementation details of the mask mechanism can be found in Appendix A.3.

Personalized Node Decoding. At each time step, the decoder calculates the probability of each candidate node and selects the node with the maximum probability as the routing node. The whole decoding process can be described by the conditional probabilities calculated by the chain rule shown in Equation 9.

$$\begin{aligned} p(\boldsymbol{\pi}|s;\theta) &= \textstyle\prod_{j=1}^{n} p(\pi_j|s, \pi_{1:j-1}, w; \theta) \\ &= \textstyle\prod_{j=1}^{n} p(\pi_j|f(s,\theta_e), \pi_{1:j-1}, w; \theta_d), \end{aligned} \tag{9}$$

where s is the problem instance, and θ represents all trainable parameters. $f(s,\theta_e)$ is the dynamic graph encoder and θ_d is the trainable parameter of the decoder. It is worth mentioning that, unlike route planning, where the routing objective function is the same for all workers, e.g., minimizing the total travel distance. In route prediction, different workers have different goals (or objective functions) in route selection, and a worker w's preference in route choices is often unobservable to the service provider. Therefore, it is necessary to model the worker in the decoder, and to learn the personalized decision preference from massive historical data.

Here we use a RNN to embed the already generated outputs $\boldsymbol{\pi}_{1:j-1}$ into a state vector $\mathbf{m}_{j-1}$. The output of $f(s,\theta_e)$ is node embedding matrix $\mathbf{H}$. The concatenation of the worker's embedding (a learnable vector) and his profile features (i.e., average speed, level, maximal load) $\mathbf{w} \in \mathbb{R}^{d_w}$ is utilized to represent a worker:

$$p(\boldsymbol{\pi}|s;\theta) = \textstyle\prod_{j=1}^{n} p(\pi_j|\mathbf{H}, \mathbf{m}_{j-1}, \mathbf{w}; \theta_d). \tag{10}$$

During the decoding process, the output node at each step should be chosen from the input graph without duplication. To meet the above requirement, we design a graph-based recurrent decoder. As shown in Figure 4, the decoding process happens recurrently, the core idea is to find the most possible neighbor of the last outputted node as the next-visit node at each decoding step, and the selected node is fed into the RNN to record the current decoding status to help the next selection. Specifically, we concatenate $\mathbf{m}_{j-1}$ and $\mathbf{w}$ to form an attention query, and treat the embeddings of candidates (i.e., neighbors of the latest output node $\mathcal{N}_{\pi_{j-1}}$) as the attention key, to produce the output probability distribution over the candidate nodes. In the implementation, we mask the nodes in the masked node set $\mathcal{V}_{mask}^j$:

$$u_i^j = \begin{cases} \mathbf{q}^T \tanh(\mathbf{W}_7 \mathbf{h}_i + \mathbf{W}_8[\mathbf{m}_{j-1}; \mathbf{w}]) & \text{if } i \notin \mathcal{V}_{mask}^j \\ -\infty & \text{otherwise,} \end{cases} \tag{11}$$

where $\mathbf{W}_7 \in \mathbb{R}^{d_h \times d_h}$, $\mathbf{W}_8 \in \mathbb{R}^{(d_h+d_w)\times d_h}$ and $\mathbf{q} \in \mathbb{R}^{d_h}$ are learnable parameters. Finally, we get the output probability p_i^j of node i at step j by using a softmax layer:

$$p_i^j = p\left(\pi_j = i \mid s, \pi_{1:j-1}; \theta\right) = \text{softmax}(u_i^j) \tag{12}$$

The attention score can be regarded as a pointer that points to a node's neighbor ($\pi_j = \arg\max_k p_k^j$), because it directly indicates the output probability of an input node. By leveraging the graph structure in the decoding process, some unlikely solutions will be removed from the solution space, thus avoiding extremely unreasonable outputs. Moreover, unlike previous works that simply concatenates the worker's features with a task's features before encoder [3, 16], we directly introduce the worker's personalized information into the decoder, which can avoid the personalized information loss during the encoding and therefore lead to a more accurate prediction.

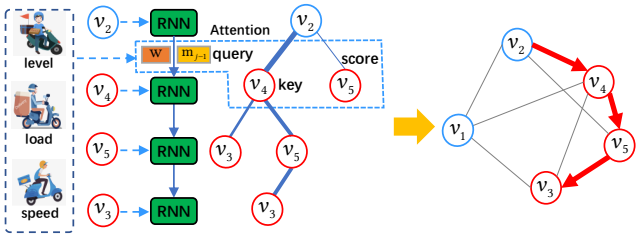

Figure 4: Illustration of Decoding Process.

3.4 Model Training and Prediction

In this part, we first introduce how the sample is constructed, then introduce the training and prediction process of Graph2Route.

Sample construction. We construct a sample when a task is finished or is dispatched to the worker. At the sample time t, we gather the finished tasks and all unfinished tasks as nodes in the input ST-graph. In the food delivery service, tasks of a worker are not settled from the beginning. Rather, they are revealed over time because the platform can continuously dispatch new tasks to the worker. In that case, new coming task at t' can change the worker's previous decisions at t, making observations after t' inaccurate [3, 17] as label for the sample at time t. Therefore, we choose the route observations between t and t' as the label information when training the model, where t' is the dispatch time of the first coming task after t. Take Figure 3 as an example, where the task's finish- or dispatch-event

Algorithm 1 Graph2Route.

Input: Input Graph $\mathcal{G}_t^w = (\mathcal{V}_t, \mathcal{E}_t, \mathbf{X}_t, \mathbf{E}_t)$ at time t of worker w.
Output: Output predicted service route $\hat{\boldsymbol{\pi}}_{t:}$.

```
// Dynamic ST-Graph Encoder
for l = 1, ..., L do // Spatial-Correlation Encoding
    Update node embedding by Eq. (6);
    Update edge embedding by Eq. (7);
end for
// Carry on the decision context though GRU
Update H_t according to Eq. (8);
// Graph-based Personalized Route Decoder
π̂_{t:} ← [];
for j = 1, ..., |V_t^U| do
    Mask infeasible nodes in V_{mask}^j;
    Calculate the output probability p_i^j by Eq. (12);
    Select output node π_j = arg max_k p_k^j;
    Append π_j into π̂_{t:};
end for
return π̂_{t:};
```

of a worker are depicted in the timeline. For the sample at t_1, its input contains task $\{1, 2, 3\}$, and the label $\boldsymbol{\pi}_{t_1:t_1'} = \boldsymbol{\pi}_{t_1:t_2} = [2]$.

Training. The decoding process predicts the probability of each feasible location the worker will visit next, which can be regarded as a multi-class classification problem. Therefore, the cross-entropy loss function is utilized to calculate the loss of the model, which is formulated as:

$$\mathcal{L} = -\sum_{w \in \mathcal{W}} \sum_{t \in \mathcal{T}} \sum_{i \in \boldsymbol{\pi}_{t:t'}} y_i \log(p(y_i|\theta)), \tag{13}$$

where $\mathcal{W}$ is the set of workers, $\mathcal{T}$ is the set of the total sampling time steps, y_i is the order of task i in the label of that sample, and $p(y_i|\cdot)$ is the predicted probability of task i predicted by the model.

Prediction. When used for predictions, the model takes all the unfinished tasks of a given time as input, and predicts the whole service route, supposing no tasks are coming in the future. Overall, the whole process of Graph2Route is illustrated in Algorithm 1.

4 EXPERIMENTS

In this section, we present both online and offline experiments to demonstrate the effectiveness of our model. The setting includes a variety of datasets, compared methods, and evaluation metrics.

4.1 Offline Experiments

4.1.1 Datasets. To show the generality of our model, we utilize two real-world datasets which belong two P&D service types:

Food Pick-up and Delivery Data. (Food-PD for short) Food-PD is online food pick-up and delivery data[1] collected by Eleme[2], accumulating millions of food delivery data every day. The dataset contains the pick-up and delivery records of 916 workers from 28 days in Dalian, China. In this dataset, we aim to predict drivers' future pick-up and delivery routes by treating drivers as workers and a pick-up/delivery task of an order as one node in our method. Detailed statistics is shown in Table 2.

Table 2: Statistics of the Datasets. (Abbr.: ANUT for Average number of unfinished tasks.)

Type	Time Range	City	ANUT	#Workers	#Samples
Food-PD	02/02/2020 - 02/29/2020	Dalian	4	916	166,026
Logistics-P	03/29/2021 - 05/27/2021	Shanghai	9	2,344	208,202

Logistics Pick-up Data. (Logistics-P for short) Logistics-P is package pick-up data collected from Cainiao[3], one of the largest logistics platforms in China. The dataset contains the pick-up records of 2344 couriers from Mar. 29, 2021 to May. 27, 2021 (60 days) in Shanghai, China. In this scenario, couriers are expected to visit customers' locations and pick up their packages. In this dataset, we aim to predict couriers' future pick-up routes by treating couriers as workers and a pick-up order as a task in our method.

4.1.2 Baselines. We implement some basic methods, machine learning methods, and state-of-the-art deep models for comparison. Note that implementations in Logistics-P do not have the pick-up then delivery constraint.

Basic methods, including i) *DisGreedy* generates the prediction route by sorting the distances step by step. ii) *TimeRank* generates the prediction route by sorting the remaining time, i.e., the promised time minus the current time. iii) *OR-Tools* [1] adopts a heuristic strategy to search the route with the minimum travel distance as worker's future route.

Machine learning method, including *OSquare* [20], which is a point-wise ranking method that trains a traditional machine learning model (i.e., LightGBM [6]) to output the next location at one step, and the whole route is generated recurrently (Details of the algorithm can be found in Appendix A.2).

Deep models, including i) *DeepRoute* [17], which is a deep model equipped with a Transformer encoder and an attention-based decoder to rank a courier's all unpicked-up packages. Note that DeepRoute cannot handle the pick-up then delivery route prediction problem, so we add the mask mechanism to its decoder to satisfy the route constraints in the Food-PD dataset. ii) *DeepRoute+* [16]. DeepRoute+ models the decision preference of workers by adding an encoding module that takes the courier's latest pick-up route into account. iii) *FDNET* [3], which utilizes the LSTM and a Pointer Network to predict the driver's route in the food delivery system. It also introduces the arrival time prediction as the auxiliary task to improve the accuracy of route prediction.

For deep models, hyperparameters are tuned using the validation set, we train, validate, and test the model at least five times and the average performance is reported. Details of the parameters can be found in Appendix A.1. And the code is available at https://github.com/wenhaomin/graph2route.

4.1.3 Metrics. We introduce a comprehensive indicator system to evaluate the performance from both local and global perspective. It is worth mentioning that the label length may be shorter than the prediction length, because of the uncertainty brought by new

[1] https://tianchi.aliyun.com/competition/entrance/231777/information
[2] https://www.ele.me/
[3] https://www.cainiao.com/

coming tasks (as introduced in Section 3.4). Formally, we have the prediction $\hat{\boldsymbol{\pi}} = [\hat{\pi}_1, \ldots, \hat{\pi}_n]$ and the label $\boldsymbol{\pi} = [\pi_1, \ldots, \pi_m]$, where $m \leq n$ and $\text{set}(\boldsymbol{\pi}) \subseteq \text{set}(\hat{\boldsymbol{\pi}})$. Let $O_{\boldsymbol{\pi}}(i)$ and $O_{\hat{\boldsymbol{\pi}}}(i)$ be the order of node i in the label and prediction route, respectively. One can evaluate the route prediction performance by the following metrics:
KRC: Kendall Rank Correlation [7] is a statistical criterion to measure the ordinal association between two sequences. Given any node pair (i, j), it is said to be concordant if both $O_{\hat{\boldsymbol{\pi}}}(i) > O_{\hat{\boldsymbol{\pi}}}(j)$ and $O_{\boldsymbol{\pi}}(i) > O_{\boldsymbol{\pi}}(j)$ or both $O_{\hat{\boldsymbol{\pi}}}(i) < O_{\hat{\boldsymbol{\pi}}}(j)$ and $O_{\boldsymbol{\pi}}(i) < O_{\boldsymbol{\pi}}(j)$. Otherwise, it is said to be discordant. To calculate this metric, nodes in the prediction are first divided into two sets: i) nodes in label $\mathcal{V}_{in} = \{\hat{\pi}_i | \hat{\pi}_i \in \boldsymbol{\pi}\}$, and ii) nodes not in label $\mathcal{V}_{not} = \{\hat{\pi}_i | \hat{\pi}_i \notin \boldsymbol{\pi}\}$. We know the order of items in $\mathcal{V}_{in}$, but it is hard to tell the order of items in $\mathcal{V}_{not}$, still we know that the order of all items in $\mathcal{V}_{in}$ are ahead of that in $\mathcal{V}_{not}$. Therefore, we compare the nodes pairs $\{(i, j) | i, j \in \mathcal{V}_{in} \text{ and } i \neq j\} \cup \{(i, j) | i \in \mathcal{V}_{in} \text{ and } j \in \mathcal{V}_{not}\}$. In this way, the KRC is defined as:

$$\text{KRC} = \frac{N_c - N_d}{N_c + N_d}, \tag{14}$$

where N_c is the number of concordant pairs, and N_d is the number of discordant pairs.
ED: Edit Distance [11] (ED) is an indicator to quantify the dissimilarity of two sequences, by counting the minimum number of required operations to transform one sequence into another.
LSD and **LMD**: The Location Square Deviation (LSD) and the Location Mean Deviation (LMD) measure the degree that the prediction deviates from the label, formulated as:

$$\begin{aligned} \text{LSD} &= \frac{1}{m} \sum_{i=1}^{m} (O_{\pi}(\pi_i) - O_{\hat{\pi}}(\pi_i))^2 \\ \text{LMD} &= \frac{1}{m} \sum_{i=1}^{m} |(O_{\pi}(\pi_i) - O_{\hat{\pi}}(\pi_i))|. \end{aligned} \tag{15}$$

HR@k: Hit-Rate@k is used to quantify the similarity between the top-k items of two sequences. It describes how many of the first k predictions are in the label, which is formulated as follows:

$$\text{HR@}k = \frac{\hat{\boldsymbol{\pi}}_{[1:k]} \cap \boldsymbol{\pi}_{[1:k]}}{k}. \tag{16}$$

ACC@k: Compared with HR@k, ACC@k is a more strict measurement to calculate the local similarity of two sequences. It answers the following question: Is the route composed of the first k predictions exactly the same as the label?

$$\text{ACC@}k = \prod_{i=0}^{k} \mathbb{I}(\hat{\pi}_i, \pi_i), \tag{17}$$

where $\mathbb{I}(\cdot)$ is the indicator function, and $\mathbb{I}(\hat{\pi}_i, \pi_i)$ equals 1 if $\hat{\pi}_i == \pi_i$ else 0.

In summary, KRC, ED, LSD, and LMD measure the overall similarity of the predicted route and the label route, while HR@k and ACC@k calculate thei similarity from the local perspective. Higher KRC, HR@k, ACC@k, and lower ED, LSD, LMD mean better performance of the algorithm.

4.1.4 Results. The predicting difficulty rises along with the number of unfinished tasks, since more tasks mean greater decision space and uncertainty. As a result, we evaluate models under the different number of tasks. The results on both datasets show that our Graph2Route model achieves promising performance improvements compared with all the baselines. On Food-PD (length 0-11), it improves the performance of the best baseline by 7.9% at ACC@3, by 12.5% at LMD, and by 8.9% at ED. Since the model serves millions of workers and billions of customers, a small increase in the value of the above metrics means a huge improvement of the model and can generate enormous economic benefits.

TimeRank and DisGreedy perform poorly since they only utilize the time or distance information. OSquare performs better than TimeRank and DisGreedy. Not only because of the superiority of the learning model architecture, but also the spatial-temporal information it incorporates. OSquare outputs the predicted route step by step. When the length of routes becomes longer, the performance drops because of the error accumulating problem.

In both scenarios, DeepRoute and DeepRoute+ outperform OSquare. The goal of OSquare is to maximize the output probability of the next location, while DeepRoute and DeepRoute+ aim to maximize the probability of the entire predicted route. DeepRoute and DeepRoute+ can capture the contextual information better than OSquare since they can capture more complex spatial-temporal relationships by the Transformer-like encoder. Moreover, DeepRoute+ models each courier's decision preference in the encoding process compared with DeepRoute, which contributes to a performance improvement of the route prediction.

Performances of all methods in Food-PD are better than Logistics-P. Mainly because the average length of unfinished tasks is shorter in food delivery, reducing the difficulty of finding the right route in a smaller solution space. Learning-based models can capture dynamic changes of the spatial-temporal constraints and performs better than non-learning-based models. FDNET, DeepRoute, DeepRoute+, and Graph2Route are deep models based on encoder-decoder architectures. We can see that Graph2Roue outperforms the others. This demonstrates the superiority of the dynamic ST-Graph encoder and graph-based personalized route decoder. Graph2Route can encode the input spatial-temporal graph and update the representations of nodes and edges by leveraging their interactions, and can capture the complex and decision context information. Furthermore, it fully utilizes the graph structure and workers' personalized information to output a more accurate service route by the graph-based personalized route decoder.

4.1.5 Component Analysis. We focus on the ablation study on the Food-PD dataset (length range 0-25). Figure 5 illustrates the results.

Firstly, we remove the spatial-correlation encoding (i.e., the GCN) in the encoder (w/o GCN). Without GCN, the performance drops significantly at all evaluation metrics. This demonstrates the powerful encoding ability of the graph-based encoder in capturing spatial-temporal information. The spatial-correlation encoding plays a key role in improving prediction performance.

Secondly, removing the temporal-correlation encoding (i.e., the GRU) in the encoder (w/o GRU) also brings a significant decrease in the model's performance. The results demonstrate that updating the node embeddings by a recurrent architecture and fully considering

Table 3: Experiment Results. Higher KRC, HR@k, ACC@k and lower LMD and LSD, ED indicates better performance.

Method	Logistics-P												Food-PD											
	$n \in (0-11]$						$n \in (0-25]$						$n \in (0-11]$						$n \in (0-25]$					
	HR@1	ACC@3	KRC	LMD	LSD	ED	HR@1	ACC@3	KRC	LMD	LSD	ED	HR@1	ACC@3	KRC	LMD	LSD	ED	HR@1	ACC@3	KRC	LMD	LSD	ED
TimeRank	26.37%	13.62%	37.76%	2.30	11.54	2.41	25.19%	12.47%	35.44%	2.45	12.84	3.19	45.02%	27.59%	60.78%	0.57	0.78	1.74	44.99%	27.57%	60.77%	0.57	0.79	1.74
DisGreedy	45.98%	26.09%	51.29%	1.72	8.45	2.01	45.43%	24.52%	49.72%	1.84	9.27	2.66	53.16%	39.92%	68.09%	0.47	0.67	1.41	53.12%	39.89%	68.09%	0.47	0.68	1.42
Or-Tools	48.59%	28.04%	54.30%	1.54	6.87	1.95	47.81%	26.26%	52.60%	1.67	7.73	2.61	54.98%	42.32%	70.37%	0.44	0.62	1.35	54.96%	42.29%	70.36%	0.44	0.62	1.35
OSquare	47.03%	24.24%	55.20%	1.52	6.01	2.05	46.32%	22.55%	53.58%	1.64	6.88	2.74	63.43%	44.79%	72.62%	0.45	0.67	1.37	63.39%	44.75%	72.59%	0.45	0.68	1.38
FDNET	49.50%	27.73%	55.75%	1.60	7.59	1.96	48.81%	25.91%	54.08%	1.72	8.38	2.62	64.69%	49.64%	75.46%	0.41	0.63	1.27	64.65%	49.60%	75.42%	0.41	0.64	1.28
DeepRoute	51.87%	28.35%	59.07%	1.42	5.98	1.96	50.88%	26.46%	57.31%	1.55	6.81	2.62	66.94%	48.67%	75.10%	0.41	0.61	1.27	66.91%	48.63%	75.08%	0.41	0.62	1.28
DeepRoute+	52.03%	28.75%	59.80%	1.39	5.73	1.94	51.14%	26.87%	58.09%	1.52	6.54	2.60	67.19%	49.77%	75.76%	0.40	0.60	1.24	67.16%	49.73%	75.75%	0.40	0.61	1.24
Graph2Route	**52.53%**	**29.25%**	**61.22%**	**1.34**	**5.21**	**1.92**	**51.56%**	**27.28%**	**59.45%**	**1.46**	**6.02**	**2.58**	**67.97%**	**53.72%**	**78.68%**	**0.35**	**0.51**	**1.13**	**67.92%**	**53.68%**	**78.67%**	**0.36**	**0.52**	**1.13**
Improvement	1.0%	1.7%	2.4%	3.6%	9.1%	1.0%	0.8%	1.5%	2.3%	3.9%	8.0%	0.8%	1.2%	7.9%	3.9%	12.5%	15.0%	8.9%	1.1%	7.9%	3.9%	10.0%	14.8%	8.9%

temporal decision contexts also contribute a lot to the performance improvement.

Lastly, we remove the worker's personalized information in the decoding process (w/o worker Info). Results show that introducing the worker's personalized information into the decoder brings an improvement, indicating that the personalized decoder can efficiently incorporate workers' personalized information during decoding and lead to a more accurate prediction.

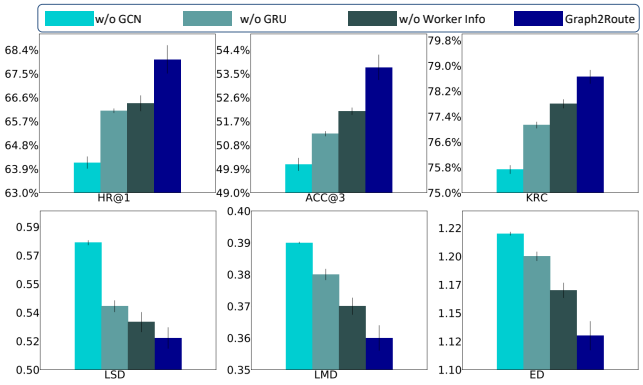

Figure 5: Component Analysis.

4.2 Online A/B Test

Motivated by the promising offline results, we deploy our model on the Cainiao logistics platform to test its performance online. Specifically, we first choose the package pick-up arrival time prediction [2] (i.e., to predict the courier's arrival time of a package's location) as the downstream task by feeding the route prediction results into the task. From the results of the A/B test, Graph2Route gains 18.7% improvement in Mean Absolute Error (MAE) of arrival time prediction, the customer complaint and inquiry rate decreased by around 20% because of the accurate arrival time prediction. Furthermore, we incorporate the route predictions into the dispatch system. From the results of the A/B test, Graph2Route helps to reduce over 100 thousand mis-dispatched packages (i.e., the package is dispatched to an inappropriate courier) each month, saving the platform millions of costs for re-allocating those mis-dispatched packages. Overall, our model brings significant improvement in user satisfaction and cost-saving.

4.3 Case Study

To analyze the performance of Graph2Route more intuitively, We provide an empirical case study shown in Figure 6.

In case 1, DeepRoute+ outputs unreasonable route while Graph2Route outputs a more reasonable route, which is also the same as the courier's actual route. Because of the sequence-based architecture, DeepRoute+ cannot avoid decoding extremely unreasonable routes. In this case, the courier visits B first then E, which is the farthest location from B. In contrast, Graph2Route introduces the graph structure in the encoding and decoding process at each time step. In this case, since E is the farthest location from B, it has no connections with B, therefore the decoder in Graph2Route will not output E as B's next location. In summary, Graph2Route can filter unreasonable solutions and reduce the solution space, so that the rationality of the prediction is improved.

In case 2, Graph2Route can provide more stable predictions at nearby time steps. In this case, both DeepRoute+ and Graph2Route provide the right route prediction at 9:05, and no new task was dispatched to the courier between 9:05 and 9:10, so the decision-making environment does not change drastically. Intuitively, the predicted route at 9:10 should not change or at least share some similarities with the one at 9:05. However, it can be seen that the predicted route at 9:10 given by DeepRoute+ is totally different from 9:05, which is not aligned with the courier's real route. In contrast, the predicted route at 9:10 given by Graph2Route is the same as the one at 9:05 and the courier's actual route. By introducing the decision context, Graph2Route can leverage the decision-making information at 9:05 into the prediction at 9:10, resulting in a more stable and accurate prediction at 9:10.

5 RELATED WORK

The rapid growth of trajectory data generated from massive workers makes data-driven methods an effective way to accurately predict their future routes. In the online food delivery system, OSquare [20] is the first work that predicts courier's pick-up and delivery route. It first converts the route prediction problem into a next-location prediction problem, trains a traditional machine learning model, and then generates the whole route recurrently. However, the performance of OSquare is limited due to the model capacity of the machine learning method. Besides, predicting in an autoregressive way may introduce the error propagation problem. To solve the above challenges, FDNET [3] proposes a deep model which utilizes

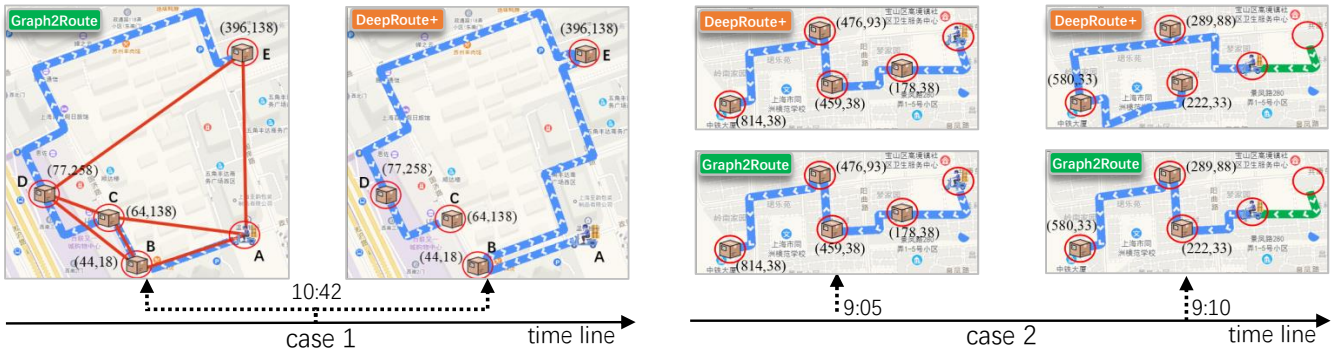

Figure 6: Case Study. The blue lines are predicted routes. In case 1, the red lines show the underlying graph structure. In case 2, the green lines are historical routes. We list two features $(x_i^{Dis}, x_i^{PT} - t)$ nearby an unfinished task i at time t, where x_i^{Dis} (meters) is the distance between the task and the courier's current location, $x_i^{PT} - t$ (minutes) is the remaining pick-up time of the task.

a Pointer Network [13] to predict the worker's route at once. Moreover, it jointly predicts the arrival times and the future route. By introducing actual arrival times as auxiliary supervision, the model is more likely to learn workers' routing behaviors.

In the logistics field, DeepRoute [17] is the pioneer to propose a deep model to predict courier's future pick-up route. It utilizes a Transfomer-based encoder to capture the spatial-temporal correlations between unfinished tasks, and decodes the whole route recurrently by a Pointer architecture. Based on DeepRoute, DeepRoute+ [16] proposes to model couriers' personalized information for more accurate prediction. Therefore, it designs a decision preference module in the task representation layer to model the influence of different factors on couriers' decisions.

Previous sequence-based models still lack effective encoders to fully capture spatial-temporal relationships and decoder to avoid extremely unreasonable results. Therefore, in this paper, we formulate the problem from the graph perspective, and propose a dynamic graph-based model for more accurate prediction.

6 CONCLUSION

In this paper, for the first time, we formulate and solve the pick-up and delivery route prediction problem from the graph perspective, and a dynamic graph neural network, Graph2Route, is proposed to address this problem. Graph2Route utilizes a dynamic ST-Graph encoder to fully capture the spatial-temporal correlations of different tasks and the evolution of the decision-making environment. Moreover, the decoder can incorporate workers' personalized information and the underlying graph structure, leading to more accurate and reasonable predictions. Extensive experiments conducted on two real-world datasets under different P&D services demonstrate the superiority of our proposed method against the classic methods and state-of-the-art neural network methods.

ACKNOWLEDGMENTS

This work was supported by Alibaba Group through Alibaba Innovative Research Program.

REFERENCES

[1] Irwan Bello, Hieu Pham, Quoc V Le, Mohammad Norouzi, and Samy Bengio. 2017. Neural combinatorial optimization with reinforcement learning. In *ICLR*.

[2] Arthur Cruz de Araujo and Ali Etemad. 2021. End-to-End Prediction of Parcel Delivery Time with Deep Learning for Smart-City Applications. *IEEE Internet of Things Journal* (2021).

[3] Chengliang Gao, Fan Zhang, Guanqun Wu, Qiwan Hu, Qiang Ru, Jinghua Hao, Renqing He, and Zhizhao Sun. 2021. A Deep Learning Method for Route and Time Prediction in Food Delivery Service. In *KDD*. 2879–2889.

[4] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.* 9, 8 (nov 1997), 1735–1780.

[5] Wenjie Huang, Zhiwei Zhao, Xiaolong An, Geyong Min, and Jingjing Li. 2020. Dynamic Scheduling for Urban Instant Delivery with Strict Deadlines. In *ICC*. 1–6.

[6] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *NeurIPS*. 3146–3154.

[7] Maurice G Kendall. 1938. A new measure of rank correlation. *Biometrika* 30, 1/2 (1938), 81–93.

[8] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

[9] Yu-Lin Lan, Fagui Liu, Wing WY Ng, Jun Zhang, and Mengke Gui. 2020. Decomposition Based Multi-Objective Variable Neighborhood Descent Algorithm for Logistics Dispatching. *IEEE Transactions on Emerging Topics in Computational Intelligence* (2020).

[10] Yiyao Li and William Phillips. 2018. Learning from route plan deviation in last-mile delivery. *Master Thesis, MASSACHUSETTS INSTITUTE OF TECHNOLOGY* (2018).

[11] John Nerbonne, Wilbert Heeringa, and Peter Kleiweg. 1999. Edit distance and dialect proximity. *Time Warps, String Edits and Macromolecules: The theory and practice of sequence comparison* 15 (1999).

[12] Paolo Toth and Daniele Vigo. 2002. *The vehicle routing problem.* SIAM.

[13] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer Networks. In *NeurIPS*. 2692–2700.

[14] Jianxin Wang, Ming K Lim, Yuanzhu Zhan, and XiaoFeng Wang. 2020. An intelligent logistics service system for enhancing dispatching operations in an IoT environment. *Transportation Research Part E: Logistics and Transportation Review* 135 (2020), 101886.

[15] Zheng Wang, Kun Fu, and Jieping Ye. 2018. Learning to Estimate the Travel Time. In *KDD*. ACM, 858–866.

[16] Haomin Wen, Youfang Lin, Huaiyu Wan, Shengnan Guo, Fan Wu, Lixia Wu, Chao Song, and Yinghui Xu. 2022. DeepRoute+: Modeling Couriers' Spatial-Temporal Behaviors and Decision Preferences for Package Pick-up Route Prediction. *ACM Trans. Intell. Syst. Technol.* 13, 2, Article 24 (jan 2022), 23 pages.

[17] Haomin Wen, Youfang Lin, Fan Wu, Huaiyu Wan, Shengnan Guo, Lixia Wu, Chao Song, and Yinghui Xu. 2021. Package Pick-up Route Prediction via Modeling Couriers' Spatial-Temporal Behaviors. In *ICDE*. IEEE, 2141–2146.

[18] Fan Wu and Lixia Wu. 2019. DeepETA: A Spatial-Temporal Sequential Neural Network Model for Estimating Time of Arrival in Package Delivery System. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 774–781.

[19] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems* 32, 1 (2020), 4–24.

[20] Yan Zhang, Yunhuai Liu, Genjian Li, Yi Ding, Ning Chen, Hao Zhang, Tian He, and Desheng Zhang. 2019. Route prediction for instant delivery. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 3, 3 (2019), Article 124.

A IMPLEMENTATION DETAILS

In this appendix, we introduce some implementation details, including the hyper-parameters setting of Graph2Route, OSquare algorithm, and the implementation of the mask mechanism in Graph2Route.

A.1 Setting of Hyper-Parameters

The parameter settings of Graph2Route are summarized in Table 4.

Table 4: The hyper parameter setup of Graph2Route.

Hyper parameters	Description	Search space
k	k-nearest spatial-temporal neighbors	[n-1, n-2]
d_h	embedding dimension of nodes/edges	[8, 16, 32]
L	the number of GCN layers	[1, 2, 3]
d_w	embedding dimension of a worker's ID	[10, 20]
B	batch size	[64, 128]
lr	learning rate	[1e-3, 5e-4]

Note that for the setting of k, we do not set the same k for all problem instances. Instead, given a problem instance that contains n nodes, if k is set $n-1$, which means we remove the farthest neighbor for each node. Hyper-parameters are tuned using the validation set, and the best parameters for both dataset are: $k = n-1$, $d_h = 8$, $L = 2$, $d_w = 10$, $B = 64$, $lr = 1e-3$.

A.2 The OSquare Algorithm

Algorithm 2 shows the implement details of OSquare.

Algorithm 2 OSquare.

Input: features of unfinished tasks of worker w at time t: $X_t = \{\mathbf{x}_1, \mathbf{x}_2 ..., \mathbf{x}_n\}$; max number of unfinished tasks max; padding vector z.
Output: predicted pick-up route. $\boldsymbol{\pi}$

$\boldsymbol{\pi} \leftarrow []$;
for $j = 1, ..., n$ **do**
  for $m \in \boldsymbol{\pi}$ **do**
    $X_t[m] = z$; //pad locations outputted before
  end for
  $X_t^{'} \leftarrow$ concatenate$(\mathbf{x}_1, ..., \mathbf{x}_n, z_{n+1}, ..., z_{\max})$;
  $\hat{y} = LightGBM(X_t^{'})$;
  $i \leftarrow \text{argmax}_k\ \hat{y}_k$, where $k \in \{1, \ldots, n\}$ and $k \notin \boldsymbol{\pi}$;
  $\boldsymbol{\pi} \leftarrow \boldsymbol{\pi} + [i]$;
end for

A.3 Mask Mechanism

Algorithm 3 shows the implementation details of the mask mechanism in Food-PD. A Pick-up node of an order at time t is denoted by a odd number $i \in \mathcal{V}_t$, while it's corresponding delivery node is denoted by $i+1$. A mask vector $\mathcal{M}_j$ is obtained at each decoding step j according to the masked node set $\mathcal{V}^j_{mask}$ in Eq. (11). If $i \notin \mathcal{V}^j_{mask}$, then $\mathcal{M}_j[i] = False$, otherwise, $\mathcal{M}_j[i] = True$.

Algorithm 3 Mask Mechanism in Food-PD.

Input: Input nodes $\mathcal{V}_t$ and infeasible nodes in $\mathcal{V}^j_{mask} = \mathcal{V}^F_t \cup \mathcal{R}_j \cup \mathcal{V}^d_j \cup \{\mathcal{V} - \mathcal{N}_{v_{\pi_{j-1}}}\}$.
Output: Output mask vector $\mathcal{M}_j$ at decoding step step j.

// Initialize mask vector
$\mathcal{M}_j \leftarrow []$;
for $i = 1, ..., |\mathcal{V}^U_t|$ **do**
  $\mathcal{M}_j \leftarrow \mathcal{M}_j + [True]$;
end for
// Update mask vector
for $i = 1, ..., |\mathcal{V}^U_t|$ **do**
  if $i \in \{\mathcal{V}^F_t \cup \mathcal{R}_j\}$ **then**
    // If it is a delivery task
    if $i//2 == 0$ **then** $\mathcal{M}_j[i] = True$;
    else
      // If it is a pick-up task,
      // remove the mask of its delivery task.
      $\mathcal{M}_j[i+1] = False$;
    end if

  else if $i \in \mathcal{V}^d_j$ **then** $\mathcal{M}_j[i] = True$;
  else
    $\mathcal{M}_j[i] = False$;
  end if

  if $i \in \{\mathcal{V}_t - \mathcal{N}_{v_{\pi_{j-1}}}\}$ **then** $\mathcal{M}_j[i] = True$;
  end if

end for
return $\mathcal{M}_j$;