M^{2}G4RTP: A Multi-Level and Multi-Task Graph Model for Instant-Logistics Route and Time Joint Prediction

Tianyue Cai[1, 2], Huaiyu Wan[1, 3], Fan Wu[2], Haomin Wen[1, 2], Shengnan Guo[1, 3†], Lixia Wu[2], Haoyuan Hu[2], Youfang Lin[1, 3]
[1]School of Computer and Information Technology, Beijing Jiaotong University, Beijing, China
[2]Artificial Intelligence Department, Cainiao Network, Hangzhou, China
[3]Beijing Key Laboratory of Traffic Data Analysis and Mining, Beijing, China
{caitianyue, hywan, wenhaomin, guoshn, yflin}@bjtu.edu.cn,
{wf118503, wallace.wulx, haoyuan.huhy}@cainiao.com

***Abstract*—Instant-logistics (e.g., food delivery and package pick-up) is increasingly calling for Route and Time Prediction (RTP), which aims to predict both future route and arrival time of a courier's unvisited locations. Accurate RTP can greatly benefit the platform, such as optimizing order dispatching and improving user experience. Although recent years have witnessed various works for solving the RTP problem, they still suffer from the following three limitations: i) Failing to consider the high-level transfer mode of couriers between AOIs (Areas Of Interest, such as residential quarters or office buildings), which can help to build more accurate RTP. ii) Failing to simultaneously make the route and time prediction. Existing works either separately predict route/time or predict them in a two-step way. However, since route and time are strongly correlated (nearby locations in the route should have similar arrival times), jointly predicting them should be more effective. iii) The widely adopted tree-based or sequence-based architecture fails to fully encode the spatial relationship between different locations. To address the above limitations, we propose a multi-level and multi-task graph model, named M^{2}G4RTP, for instant-logistics route and time joint prediction. Specifically, we propose a multi-level graph encoder equipped with a newly-designed GAT-e encoding module to capture couriers' both high-level transfer modes between AOIs and low-level transfer modes between locations. Moreover, a multi-task decoder is presented to jointly predict the route and time at different levels. Finally, a loss weighting method based on homoscedastic uncertainty is designed to balance the two tasks adaptively. Extensive experiments on an industry-scale real-world dataset, as well as the online deployment on Cainiao Alibaba, demonstrate the superiority of our proposed model.**

***Index Terms*—Instant-Logistics, Route and Time Prediction, Multi-Level Graph, Multi-Task Learning**

I. Introduction

With the popularity of online shopping and online food ordering, instant-logistics services (e.g., food delivery and package pick-up) are undergoing explosive development. Among many research topics in the field of instant-logistics service, the service **R**oute and arrival **T**ime **P**rediction (**RTP**) is increasingly becoming a hot topic in both the academic and industry communities [1]–[4]. In the instant-logistics service, a courier usually has several locations to be visited at a certain time. RTP aims to predict the future route as well as the arrival time of the courier's all unvisited locations. Joint prediction of the route and time is of great significance for the platform. On the one hand, accurate RTP prediction can greatly alleviate users' anxiety and further improve their experience [1], [2], [5], [6]. On the other hand, RTP takes a fundamental role in the intelligent logistics platform that can benefit a large number of downstream tasks, such as order dispatching [7]–[9].

†Corresponding author.

TABLE I: Comparison between our model and related ones.

Method	Multi-level	Route/Time	Method
OSquare	×	Route Only	Tree-based
DeepRoute	×	Route Only	Sequenced-based
DeepETA	×	Time Only	Sequenced-based
Graph2Route	×	Route Only	Graph-based
FDNET	×	Route&Time (Separately)	Sequenced-based
M^{2}G4RTP	✔	**Route&Time (Jointly)**	**Graph-based**

Recent years have witnessed various works that aim to solve the route, time, or route&time prediction problems. For route prediction, DeepRoute [3] and Graph2Route [10] adopt Transformer-based and graph neural networks to predict the worker's service route, respectively. For time prediction, DeepETA [2] proposes an attention-based model to predict the arrival time of package delivery. Our work walks along the same direction with FDNET [1], which tries to give both route and time prediction. However, FDNET uses two separately trained modules to predict the route and time while failing to share any global knowledge between the two modules. Although the above works have achieved promising results, they still have the following limitations (the comparison of those methods and ours is shown in Table I):

1) Failing to consider the high-level transfer mode of couriers between AOIs. As shown in Figure 1, there are actually multi-level transfer modes for couriers, namely AOI-level and location-level. They usually tend to visit all the locations in one AOI once and then go to another AOI. In practice, couriers have their own high-level transfer modes between different AOIs. Take the courier in Figure 1 as an example, he always visits AOI A first, then visit AOI B with a large possibility among his historical service trips. To this end, incorporating such high-level transition knowledge can inherently determine the visit order of some locations, which can greatly help us to improve the performance of route and time prediction. However, to the best of our knowledge, such high-level patterns have been ignored by all existing works.

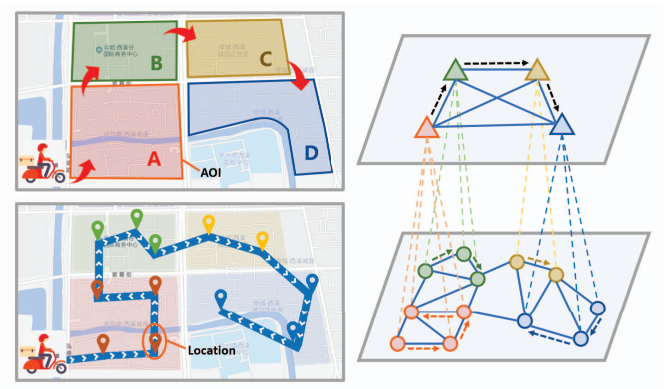


Fig. 1: Multi-Level transfer mode of courier.

2) Failing to simultaneously make the route and time prediction. Specifically, as shown in Table I, most previous methods only perform one task, i.e., route prediction or time prediction. FDNET is currently the only work that generates both route and time prediction. It utilizes a two-step architecture: first, predicting the route based on a group of input features, then generating the estimated time based on the route prediction and another group of features. However, the above architecture will naturally lead to an error accumulation problem, heavily decreasing the model's performance. Since route and time are strongly correlated (e.g., the nearby locations in a route should have similar arrival times), joint prediction introduces additional labels for the shared layers in the model, which will intuitively lead to better results. Hence, it is natural to ask for a more effective and elegant way to build a model that can jointly predict route and time.

3) Limitation of spatial correlation modeling capability. The previous works adopt the tree-based or the sequence-based architecture, limiting their ability to fully encode the spatial correlation (e.g., distance or connectivity) between different locations. Intuitively, it is a better way to build a graph, where nodes represent the feature of locations and edges represent the similarity feature between locations. Graph2Route builds an effective graph-based model. However, it still ignores the high-level transfer patterns and can only solve the route prediction problem.

To cope with the above limitations, we propose a Multi-level and Multi-task Graph model, named M^{2}G4RTP, for joint Route and Time Prediction. To address limitations 1) and 3), we design a multi-level graph-based encoder. Equipped with the AOI-level and location-level graph, the encoder can effectively learn the transfer patterns among the two levels for more accurate prediction. To address limitation 2), we propose a multi-task decoder to simultaneously predict the route and time. The joint prediction architecture enables the route and time prediction tasks to share the representation of locations and make better use of their natural relevance. However, the nature of route and time prediction is classification and regression, respectively. They are two heterogeneous tasks, and the loss function values are at different scales. To tackle this, we utilize a homoscedastic uncertainty-based loss weighting method to adaptively balance the tasks in the training process. In summary, the contributions of this work are summarized as follows:

- We are the first to study the joint prediction of route and time in instant-logistics. And a multi-level and multi-task graph model is proposed for simultaneously predicting a courier's route and arrival time.
- A multi-level graph encoder equipped with GAT-e encoding module is designed for modeling couriers' high-level transfer mode between AOIs and low-level transfer mode between locations.
- A multi-task decoder is designed to jointly predict route and time at both AOI-level and location-level. And a homoscedastic uncertainty-based loss weighting method is utilized to adaptively adjust the weight of the route loss and time loss.
- Extensive experiments on an industry-scale real-world dataset and the online deployment demonstrate the superiority of our proposed model.

II. Related Work

Recent years have witnessed various works that aim to solve the route, arrival time, or route&time prediction problem.

A. *Route Prediction*

Different from the route planning problem, route prediction in the field of instant-logistics refers to the prediction of the order in which couriers visit several locations. OSquare [4] uses a machine learning model, XGBoost, to solve the problem of delivery route prediction, which is also the first attempt to predict the route in the field of instant-delivery. DeepRoute [3] attempts to perform the route prediction with the deep learning method, which utilizes a Transformer-based encoder to encode all the unpick-up packages and an attention-based decoder to predict the next visit location step by step. Graph2Route [10] is the first attempt to use a graph-based encoder to model the spatial-temporal correlation of unpick-up packages.

It is worth noting that our work aims to predict what route the courier will take to complete the visit to all the locations by modeling various spatial-temporal features. Similar to but different from our research, plenty of route planning works have been proposed [11]–[13], which give a unified formulation of route planning problem and try to give an optimal route under different optimization objectives. In fact, the goals of the two kinds of research are different. The route planning results often differ from the real route and cannot be directly used as the prediction results.

B. Time Prediction

For time prediction, many kinds of time prediction methods are proposed in different fields of industry.

Online ride-hailing platforms like Uber have taken travel time estimation as the most important service. Some of the works are path-free methods [14]–[16], whose inputs are the origin and destination locations but without any travel route information, which is similar to our work. Some works are conducted based on a given travel route [17]–[19]. For example, CompactETA [20] models the correlation between different road segments in terms of a spatial-temporal weighted road network.

In the instant-logistics scenario, the time prediction serves in food delivery tasks or package last-mile delivery/pick-up tasks. [5] investigates three kinds of convolutional-based neural networks and assesses their performances on the parcel delivery time estimation. DeepETA [2] proposes spatial encoding and recurrent cells to capture the spatial-temporal and sequential features of the latest delivery route. And two attention-based layers are designed to indicate the most likely ETA from historical delivery routes based on the similarity of the latest route and the future destinations.

C. Route&Time Prediction

FDNET [1] is the first model trying to give both route and time prediction. FDNET is essentially a two-step model, which first utilizes a model based on LSTM and attention to predict the delivery route and then uses a Wide&Deep model to complete the time prediction based on the predicted route. FDNET is undeniably a good attempt at joint prediction, but its sequence-based encoder and two-stage training manner still make it less effective.

To sum up, researchers from various fields have come up with a large number of excellent methods to solve the problem of route, time and route&time prediction problem. However, due to different research directions or methods, the existing methods can not be directly used to solve the route&time prediction in instant-logistics, or are subject to some limitations. So that led to our work in this paper.

III. PRELIMINARY

In this section, several basic concepts and RTP problem are defined.

A. Basic Definition

Definition 1 (Location) In the instant-logistics scenario, users will place an order on the online platform. The platform will dispatch the order to the only suitable courier to complete it. The courier must visit the location of the order within the specified time to complete the delivery or pick-up on time. From the perspective of the courier, the basic feature of a location can be modeled as a triplet:

$$l = (g^l, a^l, t^{deadline}), \tag{1}$$

where g^l is the positional information (i.e., longitude and latitude) of a location, a^l refers to the AOI where the location is located. $t^{deadline}$ represent the promised arrival deadline given by the platform.

Definition 2 (AOI) Namely Area Of Interest, refers to a certain regional entity in the map. The AOI is divided according to the structure of the road network. Each AOI refers to a community, an office building, or a shopping mall. The locations to be visited by a courier belong to different AOIs. The same AOI may also contain multiple locations. The feature of an AOI can be modeled as a triple:

$$a = (id, type, g^a), \tag{2}$$

where id represents the unique ID of AOI, $type$ represents the type of AOI (community, office building, hospital, etc). In the modeling process, we focus on the visiting sequence of multiple AOIs, so we abstract an AOI as a point and take its central longitude and latitude as its geographical information g^a.

Definition 3 (Multi-Level Graph) In order to model both AOI-level and location-level transfer patterns, we treat both locations and AOIs as nodes and build a multi-level graph $\boldsymbol{\mathcal{G}}$:

$$\boldsymbol{G}^l = (\mathcal{V}^l, \mathcal{E}^l, \boldsymbol{X}^l, \boldsymbol{E}^l), \tag{3}$$

$$\boldsymbol{G}^a = (\mathcal{V}^a, \mathcal{E}^a, \boldsymbol{X}^a, \boldsymbol{E}^a), \tag{4}$$

$$\boldsymbol{\mathcal{G}} = (\boldsymbol{G}^l, \boldsymbol{G}^a, \mathcal{E}^{la}) \tag{5}$$

where $\boldsymbol{G}^l$ represents the location graph, $\mathcal{V}^l = \{l_1, l_2, ..., l_n\}$ represents the locations to be visit by a courier, $\mathcal{E}^l = \{(i,j) \mid l_i, l_j \in \mathcal{V}^l\}$ is the set of edges. $\boldsymbol{X}^l \in \mathbb{R}^{n \times d_v^l}$ is the node feature (e.g., longitude and latitude, promised arrival time, distance from courier), where d_v^l is the node feature dimension, n is the number of locations to be visited by the courier. $\boldsymbol{E}^l \in \mathbb{R}^{n \times n \times d_e^l}$ is the edge features (e.g., distance between two locations), where d_e^l is the edge feature dimension. $\boldsymbol{G}^a$ represents the AOI graph, $\mathcal{V}^a = \{a_1, a_2, ..., a_m\}$ represents the AOIs that the courier is going to visit, $\mathcal{E}^a = \{(i,j) \mid a_i, a_j \in \mathcal{V}^a\}$ is the edge set. $\boldsymbol{X}^a \in \mathbb{R}^{m \times d_v^a}$ and $\boldsymbol{E}^a \in \mathbb{R}^{m \times m \times d_e^a}$ is the feature of nodes and edges respectively, where d_v^a and d_e^a are the node and edge feature dimension, m is the number of AOI nodes in the AOI graph. $\boldsymbol{\mathcal{G}}$ represents the multi-level graph, where $\mathcal{E}^{la} = \{(i,j) \mid l_i \in \mathcal{V}^l, l_j \in \mathcal{V}^a\}$ is the edges that connect locations and their corresponding AOIs.

Definition 4 (Route) The route in this paper is defined as a sequence of locations/AOIs. The sequence implies the order of locations/AOIs to be visited by the couriers:

$$\boldsymbol{\pi}(\mathcal{V}^l) = [\pi_1^l, ..., \pi_n^l], \tag{6}$$

$$\boldsymbol{\pi}(\mathcal{V}^a) = [\pi_1^a, ..., \pi_m^a], \tag{7}$$

where $\pi_j^l \in \{1, ..., n\}$ and $\pi_j^a \in \{1, ..., m\}$. If $j \neq j'$, then $\pi_j^* \neq \pi_{j'}^*$, $* \in \{a, l\}$. For example, π_j^l indicates that π_j^l-th location is at the j-th position of the route.

Definition 5 (Time) We define the target of time prediction as the gap between the current time and arrival time at location/AOI, which can be formulated as follows:

$$y_i^l = t_i^{l,arri} - t, \tag{8}$$

$$y_j^a = t_j^{a,arri} - t, \tag{9}$$

where $t_i^{l,arri}$, $t_j^{a,arri}$ refer to the actual arrival time at location l_i and AOI a_j respectively. Specifically, we take the time when a courier arrivals the first location in the AOI as its AOI arrival time.

B. Problem Definition

Our target is to predict the route as well as arrival time for each location, formulated as:

$$\text{Target} = ([\pi_1^l, \pi_2^l, ..., \pi_n^l], [y_1^l, y_2^l, ..., y_n^l]). \tag{10}$$

We formulate the RTP request of courier u at time t as a triple $\boldsymbol{q} = (u, t, \boldsymbol{x}^g, \mathcal{V}^l)$, where $\boldsymbol{x}^g$ refer to other global features that may affect route and time prediction (e.g. weather, weekday, etc). Recall that $\mathcal{V}^l$ is the unvisited locations at the request time. Given the multi-level graph $\boldsymbol{\mathcal{G}}$, our goal is to build a model $\mathcal{F}$ that can map the input query to route and time of the unvisited locations:

$$\mathcal{F}(\boldsymbol{q}, \boldsymbol{\mathcal{G}}) \mapsto ([\pi_1^l, \pi_2^l, ..., \pi_n^l], [y_1^l, y_2^l, ..., y_n^l]). \tag{11}$$

Table II lists the notations used throughout the paper.

TABLE II: Summary of symbol notations

Notation	Definition
l	Location
a	AOI
$\mathcal{V}^l$	The set of locations to be visited of courier u at time t
$\mathcal{V}^a$	The set of AOIs to be visited of courier u at time t
$\boldsymbol{\mathcal{G}}$	The multi-level graph of locations and AOIs
$\boldsymbol{q}$	The RTP request
$\boldsymbol{\pi}(\mathcal{V}^a)$	The route of AOIs
$\boldsymbol{\pi}(\mathcal{V}^l)$	The route of locations
y^a	The arrival time of AOI
y^l	The arrival time of location

IV. Methodology

In this section, we introduce the proposed **M**ulti-level and **M**ulti-task **G**raph-based **R**oute and **T**ime **P**rediction model, namely M^{2}G4RTP, to tackle the RTP problem in instant-logistics service. We first give an overview framework of M^{2}G4RTP. Then we specify the structure of multi-level encoder and multi-task decoder, respectively. Details of the model's training process are presented at last.

A. Framework of M^{2}G4RTP

In the process of planning visiting routes, the locations and AOIs will form a global view in the courier's mind, in which spatial-temporal factors such as distance and deadline are taken into account. Based on the above understanding, we design an encoder-decoder structure model to simulate the decision-making process of the courier.

Specifically, Figure 2 illustrates the proposed M^{2}G4RTP, which consists of a multi-level graph encoder and multi-task decoders at two levels. The multi-level graph encoder takes the multi-level graph as input and learns the location/AOI representations by modeling their inter- and intra-relationships between locations and AOIs. Then, the multi-task decoder decodes the AOI-visiting route and arrival time based on the courier's current state and the AOI representations, then further decodes the location-visiting route and arrival time based on the courier's current state, location representation, and AOI-level guidance information. Finally, the M^{2}G4RTP is trained in a multi-task manner, where the loss weight assignment method based on homoscedastic uncertainty is used to balance the loss from the time and route prediction tasks of two levels.

B. Multi-level Graph Encoder

To better model the process by which couriers perceive locations and AOIs, we built a location graph $\boldsymbol{G^l}$ and an AOI graph $\boldsymbol{G^a}$. Each location or AOI is considered as a node in the graph.

Node Feature. Given a location (node) $l_i \in \mathcal{V}^l$, the node feature vector $\boldsymbol{x}_i^l$ contains spatial and temporal information related to the location and is formulated as follows:

$$\boldsymbol{x}_i^l = (x_i^{l,geo}, x_i^{l,dis}, x_i^{l,id}, x_i^{l,type}, x_i^{l,dead}, x_i^{l,acc}, x_i^{l,dead} - t), \tag{12}$$

where $x_i^{l,geo}$ is the geographical coordinate of the location, $x_i^{l,dis}$ is the distance between location l_i and the courier, $x_i^{l,id}$ and $x_i^{l,type}$ are the ID and type of AOI where the location belongs to, $x_i^{l,acc}$ is the accept time of the location's order, $x_i^{l,dead}$ is the arrival deadline of the location.

Given an AOI (node) $a_i \in \mathcal{V}^a$, the node feature vector $\boldsymbol{x}_i^a$ contains spatial and temporal information related to the AOI and is formulated as follows:

$$\boldsymbol{x}_i^a = (x_i^{a,geo}, x_i^{a,dis}, x_i^{a,id}, x_i^{a,type}, x_i^{a,dead}, x_i^{a,dead} - t), \tag{13}$$

where $x_i^{a,geo}$ is the central geographical coordinate of the AOI, $x_i^{a,dis}$ is the distance between the courier and the center of AOI, $x_i^{a,id}$ and $x_i^{a,type}$ are the ID and type of AOI, $x_i^{a,dead}$ is the earliest arrival deadline for all locations in the AOI.

Edge Feature. Given an edge $(i,j) \in \mathcal{E}^l$, the edge feature vector $\boldsymbol{e}_{i,j}^l$ contains spatial and temporal information related to the task and is formulated as follows:

$$\boldsymbol{e}_{i,j}^l = (e_{i,j}^{l,dis}, e_{i,j}^{l,gap}, e_{i,j}^{l,con}), \tag{14}$$

where $e_{i,j}^{l,dis}$ is the Euclidean distance between location l_i and l_j, $e_{i,j}^{l,gap}$ is the gap time between two arrival deadlines of location l_i and l_j, $e_{i,j}^{l,con}$ is the connectivity between two locations, which is defined as follows:

$$e_{i,j}^{l,con} = \begin{cases} 1, \text{ if } i \text{ and } j \text{ are } k\text{-nearest spatial neighbors}, \\ 1, \text{ if } i \text{ and } j \text{ are } k\text{-nearest temporal neighbors}, \\ 1, \text{ if } i = j, \\ 0, \text{ others}. \end{cases} \tag{15}$$

We define the k-nearest spatial neighbors according to the distance between nodes and the temporal neighbors according to the time gap between deadlines of each node. Similarly, Given an edge $(i,j) \in \mathcal{E}^a$, its feature can be formulated as follow:

$$\boldsymbol{e}_{i,j}^a = (e_{i,j}^{a,dis}, e_{i,j}^{a,gap}, e_{i,j}^{a,con}). \tag{16}$$

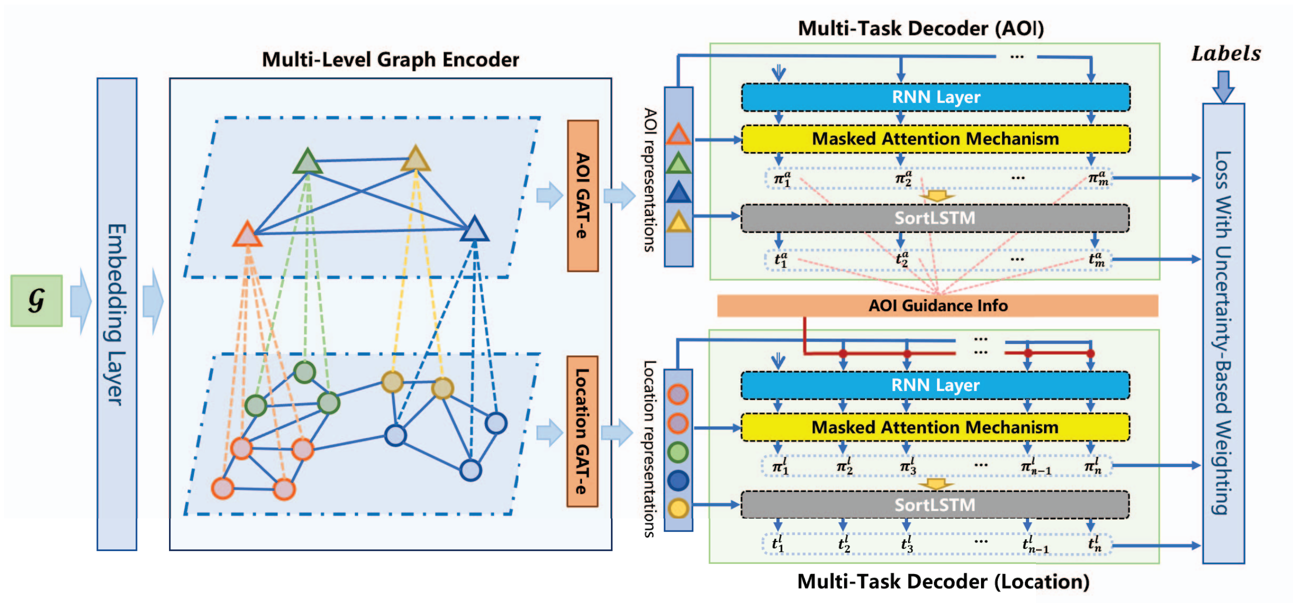


Fig. 2: Architecture of M^{2}G4RTP, which consists of a multi-level graph encoder and multi-task decoders of two levels.

Global Feature.

In addition to the spatial-temporal information of locations or AOIs, the courier's own attributes (working hours, driving speed, attendance, etc.), weather or weekday also affect the visiting route and time. The above features are collectively referred to as global features, which can be formulated as follows:

$$\boldsymbol{x}^g = (x^g_T, x^g_v, x^g_{attend}, x^g_{weather}, x^g_{weekday}), \tag{17}$$

where x^g_T, x^g_v, x^g_{attend} refer to courier's average working hours per day, average driving speed, and attendance in last two months, respectively. $x^g_{weather}$ and $x^g_{weekday}$ refer to code name of weather and weekday.

Multi-Level GAT-e Encoding Module. Before putting all node or edge features into the encoder, we use two different methods to embed the original features. For discrete features, such as AOI ID and AOI type, we apply an embedding layer to project these discrete values to the d_{disc}-dimensional vector space. For continuous features, such as coordinates or time, we apply a linear layer to project them to the d_{conti}-dimensional vector space. We concatenate the two kinds of vectors into new node or edge features as the input of the encoder. Global features will be encoded into global feature vectors by the same way and concatenated with the node feature vector. The above process can be formulated as follows (take the node and edge feature in the location graph as an example):

$$\overline{\boldsymbol{X}}^l = (\boldsymbol{W}_v\boldsymbol{X}^l_{conti} + \boldsymbol{b}_v) \oplus \mathrm{Embed}(\boldsymbol{X}^l_{disc}) \in \mathbb{R}^{n\times d_l}, \tag{18}$$

$$\overline{\boldsymbol{E}}^l = (\boldsymbol{W}_e\boldsymbol{E}^l + \boldsymbol{b}_e) \in \mathbb{R}^{n\times n\times d_l}, \tag{19}$$

where $\boldsymbol{W}_v$, $\boldsymbol{W}_e$ and $\boldsymbol{b}_v$, $\boldsymbol{b}_e$ are learnable parameters. Node feature $\overline{\boldsymbol{X}}^a \in \mathbb{R}^{n\times d_a}$ and edge feature $\overline{\boldsymbol{E}}^a \in \mathbb{R}^{n\times n\times d_a}$ in AOI graph are obtained in a similar way.

Multi-level graph encoder models the process by which couriers build global view of all locations and AOIs. From the perspective of couriers, locations/AOIs that are geographically adjacent or share the same arrival deadline can be considered as neighbors. Intuitively, we can regard location/AOIs as nodes in a graph. In addition, we introduce edge features (such as distance, gap between deadlines, connectivity) to explicitly express the similarity of nodes. Therefore, we propose an improved graph attention network [21], which takes the edge feature into account and introduces a new information-passing mechanism to update the edge features, denoted as GAT-e.

We first calculate the correlation between nodes based on the mask attention mechanism. The key to this process is to calculate the attention correlation coefficient between nodes. Let $\boldsymbol{h}^{l,k}_i$ denote the embedding of location node i at layer k, and $\boldsymbol{z}^{l,k}_{i,j}$ denotes the embedding of edge (i,j). The attention coefficient $c^{l,k}_{ij}$ can be calculated as:

$$c^{l,k}_{ij} = \sigma(\boldsymbol{a}_v(\boldsymbol{W}_1\boldsymbol{h}^{l,k}_i || \boldsymbol{W}_1\boldsymbol{h}^{l,k}_j) + \boldsymbol{a}_e\boldsymbol{z}^{l,k}_{ij}), \tag{20}$$

where $\boldsymbol{W}_1 \in \mathbb{R}^{d_l\times d_l}$, $\boldsymbol{a}_v \in \mathbb{R}^{1\times 2d_l}$, $\boldsymbol{a}_e \in \mathbb{R}^{1\times d_l}$ are learnable parameters, σ denotes the LeakyReLU activation function, $||$ is the concatenation operation. Then we utilize softmax mechanism to regularize the attention coefficients of all the neighbors of node i:

$$\alpha^{l,k}_{ij} = \frac{\exp(c^{l,k}_{ij})}{\sum_{j'\in\mathcal{N}_i}\exp(c^{l,k}_{ij'})}, \tag{21}$$

where $\mathcal{N}_i$ denotes the neighbor set of AOI a_i, $j' \in \mathcal{N}_i$ if and only if $e^{con}_{i,j'} = 1$. The regularized attention coefficient $\alpha^{l,k}_{ij}$ will be used to update the node representation:

$$\boldsymbol{h}^{l,k+1}_i = \mathrm{ReLU}(\sum_{j\in\mathcal{N}_i}\alpha^l_{ij}\boldsymbol{W}_2\boldsymbol{h}^{l,k}_i). \tag{22}$$

Edge features are updated based on linear transformation:

$$\boldsymbol{z}_{ij}^{l,k+1} = \mathrm{ReLU}(\boldsymbol{W}_3 \boldsymbol{z}_{ij}^{l,k} + \boldsymbol{W}_4 \boldsymbol{h}_i^{l,k} + \boldsymbol{W}_5 \boldsymbol{h}_j^{l,k}), \quad (23)$$

where $\boldsymbol{W}_2, \boldsymbol{W}_3, \boldsymbol{W}_4, \boldsymbol{W}_5 \in \mathbb{R}^{d_l \times d_l}$ are learnable parameters.

To stabilize the learning process of attention mechanism, P independent attention mechanisms execute the transformation of Equation 24 and 25, and then their features are concatenated:

$$\boldsymbol{h}_i^{l,k+1} = \overset{P}{\underset{p=1}{\Big\|}} \mathrm{ReLU}(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^{l} \boldsymbol{W}_2 \boldsymbol{h}_i^{l,k}), \quad (24)$$

$$\boldsymbol{z}_{ij}^{l,k+1} = \overset{P}{\underset{p=1}{\Big\|}} \mathrm{ReLU}(\boldsymbol{W}_3 \boldsymbol{z}_{ij}^{l,k} + \boldsymbol{W}_4 \boldsymbol{h}_i^{l,k} + \boldsymbol{W}_5 \boldsymbol{h}_j^{l,k}). \quad (25)$$

Specially, in the last layer (K-th layer) of encoder, we replace concatenation operation with averaging, and delay applying the activation:

$$\tilde{\boldsymbol{x}}_i^{l} = \mathrm{ReLU}(\frac{1}{P} \sum_{p=1}^{P} \sum_{j \in \mathcal{N}_i} \alpha_{ij}^{l} \boldsymbol{W}_2 \boldsymbol{h}_i^{l,K}). \quad (26)$$

So far, we have obtained the encoded location representation $\tilde{\boldsymbol{X}}^{l} \in \mathbb{R}^{n \times d_l}$. In parallel, the AOI representation $\tilde{\boldsymbol{X}}^{a} \in \mathbb{R}^{n \times d_a}$ is also obtained based on the proposed GAT-e module.

C. Multi-Task Decoder

The decoder will complete both the route and time prediction in a multi-task manner for the location- and AOI-level, respectively. During the process of route decision, the courier will decide the next location/AOI step by step. In each step, the selection of the next location/AOI is essentially to calculate the conditional probabilities of all unvisited locations/AOIs under the current state. Therefore, we need a module with a recurrent structure that can model the correlation between current state and unvisited locations/AOIs in every step. Once the route is determined, the arrival time is related to the couriers' commuting time between locations/AOIs and the service time of staying at a certain location/AOI. We need another module equipped with a sorting function to arrange nodes according to the predicted route and then predict the arrival time of each node step by step.

Specially, the route and time prediction results at AOI-level are used as important guidance information for route and time prediction at location-level. This prediction paradigm of "AOI guiding Location" is similar to the idea of divide-and-conquer, which is utilized in the quicksort algorithm [22]. During the process of quicksort, all the elements to be sorted are divided into two groups, and all the elements in one group are larger than those in the other group. Similarly, locations are naturally grouped by AOIs. The guidance information from AOI-level will inform the location-level decoder of the approximate position of a location in the route and the rough scope of arrival time.

According to the above understanding, we design an RNN-based module equipped with masked attention mechanism to simulate the decision-making process of couriers. And we propose an RNN-based network with sorting function, namely SortLSTM, to calculate the arrival times. A total of four modules from two different levels work closely in a multi-task manner during both training and prediction phases. Figure 3 shows the architecture of AOI-level decoder.

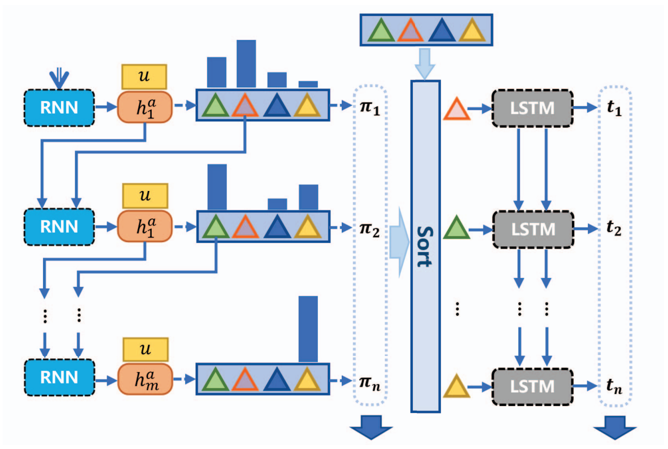


Fig. 3: Architecture of Multi-task Decoder.

AOI-level Decoder. For AOI route prediction, at each step of decoding, the decoder calculates the probability of each candidate AOI node and selects the node with the maximum probability as the routing node. The whole decoding process of courier u at time t can be described by the conditional probabilities calculated by the chain rule shown in Equation 27:

$$p(\boldsymbol{\pi}(\mathcal{V}^a) \mid \boldsymbol{q}, \boldsymbol{\mathcal{G}}; \theta) = \prod_{i=1}^{m} p\left(\pi_i^a \mid \tilde{\boldsymbol{x}}_i^a, \boldsymbol{\pi}_{1:i-1}^a, \boldsymbol{u}; \theta_d\right), \quad (27)$$

where θ denote all the learnable parameters, $\tilde{\boldsymbol{x}}^a$ is the representation of AOIs obtained from encoder, θ_d denotes the learnable parameters in decoder.

In order to get the representation of the current state at each step, we use an LSTM to aggregate the already generated AOIs $\boldsymbol{\pi}_{1:i-1}^a$ into vector $\boldsymbol{h}_{i-1}^a$. Then we concatenate the courier's embedding and his profile features $\boldsymbol{u} \in \mathbb{R}^{d_u}$ (i.e., days of work, average speed) to represent a courier:

$$p(\boldsymbol{\pi}(\mathcal{V}^a) \mid \boldsymbol{q}, \boldsymbol{\mathcal{G}}; \theta) = \prod_{i=1}^{m} p\left(\pi_i^a \mid \tilde{\boldsymbol{x}}_i^a, \boldsymbol{h}_{i-1}^a, \boldsymbol{u}; \theta_d\right). \quad (28)$$

For s-th step of decoding, our goal is to find the most likely neighbor of the $(s-1)$-th output, while ensuring that no node is output repeatedly. Then the neighbor with the greatest possibility selected in the current decoding step will be used as the input of LSTM in the next decoding step to obtain a new hidden state $\boldsymbol{h}_s^a$. Specifically, we take the concatenation of $\boldsymbol{h}_{s-1}^a$ and $\boldsymbol{u}$ as the query, and the neighbors of the node output in the previous step as the key to calculate the probability distribution among all the neighbors:

$$o_s^{a,j} = \begin{cases} \boldsymbol{v}^\top \tanh\left(\boldsymbol{W}_6 \tilde{\boldsymbol{x}}_j^a + \boldsymbol{W}_7 \left[\boldsymbol{h}_{s-1}^a \| \boldsymbol{u}\right]\right) & \text{if } j \notin \mathcal{V}_{\text{mask}}^s, \\ -\infty & \text{otherwise,} \end{cases} \quad (29)$$

where $\boldsymbol{W}_6 \in \mathbb{R}^{d_a \times d_a}$, $\boldsymbol{W}_7 \in \mathbb{R}^{(d_a+d_u)\times d_a}$ and $\boldsymbol{v} \in \mathbb{R}^{d_a}$ are learnable parameters, $j \in \mathcal{V}^s_{\text{mask}}$ if node j has been output in previous step. Finally, we get the probability of each output node in step i through softmax operation:

$$p_s^{a,j} = p\left(\pi_s^a = j \mid \tilde{\boldsymbol{x}}_j^a, \boldsymbol{\pi}^a_{1:s-1}, \boldsymbol{u}; \theta\right) = \frac{\exp(o_s^{a,j})}{\sum_{j' \notin \mathcal{V}^s_{\text{mask}}} \exp(o_s^{a,j'})}. \tag{30}$$

Then take the maximum of all probabilities in step s to form the route prediction output in step s:

$$\pi_s^a = \arg\max_j p_s^{a,j}. \tag{31}$$

The goal of our time prediction module is to fully use the results of route prediction and minimize the accumulation of bias. To solve the above problems, we propose an RNN-based network with a sorting function, namely SortLSTM.

Specifically, SortLSTM is composed of several LSTM cells, which recurrently outputs the arrival time of each AOI. The input of the i-th LSTMCell is the π_i^a-th node. In other words, the input of the SortLSTM is essentially a sequence of AOI representations sorted by the route prediction result. Besides, to help the model better sense the position of the current node in the route, we introduce the position encoding module, which is inspired by Transformer [23]:

$$\begin{aligned} \boldsymbol{p}_{(pos,2k)} &= \sin(pos/r^{2k/dim}) \\ \boldsymbol{p}_{(pos,2k+1)} &= \cos(pos/r^{2k/dim}), \end{aligned} \tag{32}$$

where pos is the AOI's position (from 1 to m) in the route. The AOI representation is concatenated with a positional encoding vector p and input in the SortLSTM. Essentially, we not only implicitly input the AOI representation into SortLSTM in the order of the route, but also explicitly concatenate position information on each step of the input, which makes the result of route prediction can be fully used in time prediction. The above process can be formulated as follow:

$$y_i^a = \text{LSTMCell}([\tilde{\boldsymbol{x}}^a_{\pi_i^a} \| \boldsymbol{p}_{pos_{\pi_i^a}}]). \tag{33}$$

It is worth mentioning that we do not force the output of $(i+1)$-th step to be greater than that of step i. This enables the time prediction module to have a certain error correction capability to avoid the error of the wrong route prediction result accumulating in the time prediction.

Location-level Decoder. In general, the location-level route/time prediction module shares the structure of the AOI-level route/time prediction module. An important difference is that the AOI-level prediction results will be used as important guidance information in the process of location-level prediction.

For location route prediction, in the s-th decoding step, we add AOI guidance information to calculate the conditional probability of each location. Specifically, we concatenate the location representation with the position encoding vector $\boldsymbol{p}_{aoi}$ and arrival time y^a_{aoi} of the AOI where the location belongs to as input, then calculate the probability distribution among all its neighbors:

$$\boldsymbol{x}^l_{in,i} = [\tilde{\boldsymbol{x}}^l_i \| \boldsymbol{p}_{aoi} \| y^a_{aoi}], \tag{34}$$

$$o_s^{l,j} = \begin{cases} \boldsymbol{v}^\top \tanh(\boldsymbol{W}_8 x^l_{in,j} + \boldsymbol{W}_9[\boldsymbol{h}^l_{s-1} \| \boldsymbol{u}]) & \text{if } j \notin \mathcal{V}^s_{\text{mask}}, \\ -\infty & \text{otherwise.} \end{cases} \tag{35}$$

Then the location route prediction output π_s^l in s-th step is calculated in the similar way of AOI route prediction.

For location time prediction, We sort all location representations based on the predicted location route, and concatenate the positional encoding vector of location to the location representation:

$$y_i^l = \text{LSTMCell}([\boldsymbol{x}^l_{in,\pi_i^a} \| \boldsymbol{p}_{pos_{\pi_i^l}}]). \tag{36}$$

D. Model Training and Prediction

We further illustrate the details of our model's training and prediction process.

The model outputs the route and time at both the location and AOI levels in a multi-task manner. That is to say, during the training process, the model is supervised by 4 losses. We regard location- and AOI-level route prediction as a multi-classification problem and apply the most commonly used cross-entropy function as their loss function. Location-level route loss $\mathcal{L}^l_{route}$ and AOI-level route loss $\mathcal{L}^a_{route}$ are calculated as follow:

$$\mathcal{L}^l_{route} = -\frac{1}{T}\sum_{t=1}^{T}(\frac{1}{n_t}\sum_{i=1}^{n_t} \pi_i^l \log(P(\hat{\pi}_i^l|\theta))), \tag{37}$$

$$\mathcal{L}^a_{route} = -\frac{1}{T}\sum_{t=1}^{T}(\frac{1}{m_t}\sum_{i=1}^{m_t} \pi_i^a \log(P(\hat{\pi}_i^a|\theta))), \tag{38}$$

where T is the number of samples in the training dataset, n_t is the number of distinct locations in the t-th sample, m_t is the number of distinct AOIs in the t-th sample.

To learn the arrival time at each location and AOI, we adopt MAE (Mean Absolute Error) to compute the location arrival time loss $\mathcal{L}^l_{time}$ and AOI arrival time loss $\mathcal{L}^a_{time}$ as follows:

$$\mathcal{L}^l_{time} = \frac{1}{T}\sum_{t=1}^{T}(\frac{1}{n_t}\sum_{i=1}^{n_t} \|y_i^l - \hat{y}_i^l\|), \tag{39}$$

$$\mathcal{L}^a_{time} = \frac{1}{T}\sum_{t=1}^{T}(\frac{1}{m_t}\sum_{i=1}^{m_t} \|y_i^a - \hat{y}_i^a\|). \tag{40}$$

Classification and regression are two heterogeneous tasks, and the loss function values are in different scales. To tackle this, we use the method of assigning task weights based on homoscedastic uncertainty [24] to balance these tasks in the training process. The calculation process of total loss $\mathcal{L}$ can be formulated as follow:

$$\begin{aligned} \mathcal{L} \; = \; & \tfrac{1}{2\sigma_1^2}\mathcal{L}^a_{route} + \tfrac{1}{2\sigma_2^2}\mathcal{L}^l_{route} + \tfrac{1}{\sigma_3^2}\mathcal{L}^a_{time} + \tfrac{1}{\sigma_4^2}\mathcal{L}^l_{time} \\ & + \log\sigma_1 + \log\sigma_2 + \log\sigma_3 + \log\sigma_4, \end{aligned} \tag{41}$$

where σ_1, σ_2, σ_3, σ_4 are learnable parameters, which will change as the training goes on.

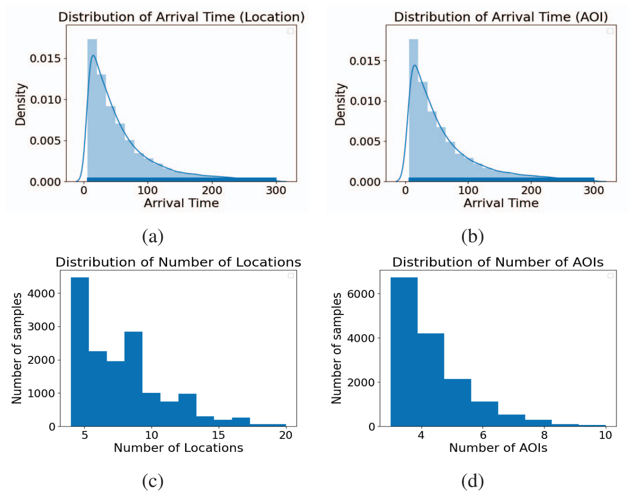


Fig. 4: Data Distribution.

When used for prediction, M^{2}GR4TP outputs two groups of route and time prediction results at the location- and AOI-level when an RTP request arrives.

V. Experiment

In this section, we first introduce the dataset used in this paper. Then we carried out extensive experiments to verify state-of-art performance and effectiveness of each component. At last, we give an empirical case study to illustrate performance of our model intuitively.

A. Dataset

The experiments are carried out on a real-world package pick-up dataset. We collected the package pick-up data in Hangzhou from July 10, 2021, to October 10, 2021. The dataset includes the order data of 8,600 AOIs and 550 couriers in Hangzhou. A single pick-up route for a single courier can contain up to 77 locations and 45 AOIs, with an average of 8.76 locations and 4.34 AOIs. In order to facilitate the training process, we selected routes with the number of locations less than 20 and the number of AOIs less than 10 as training samples. We split the 3 months into 65 days, 17 days, and 10 days as training, validation, and test sets, respectively.

Specifically, Figure 4 shows the distribution of data. Figure 4(a) and 4(b) show the actual arrival time of the locations and AOIs, averaging 59.64 and 61.68 (in minutes) respectively. Most locations/AOIs will be visited within 120 minutes. Figure 4(c) and 4(d) show how many locations/AOIs are contained in one sample, with average of 7.64 locations and 4.08 AOIs.

Additional data analysis is carried out to confirm that the high-level transfer mode of couriers between AOIs does exist, that is, couriers are used to completing all orders in a same AOI before going to another. We sort the orders of each courier in a day by the completion time, so as to get the location and the AOI sequence. We find that the average number of transfers of couriers between locations is 50.97, while the average number of transfers between AOIs is 6.20. This phenomenon shows that the transfers of couriers between AOIs are far fewer than that between locations. Couriers tend to complete most or all orders within an AOI.

B. Baselines

We implement a basic method, as well as some state-of-the-art deep models in different scenarios (food delivery and last-mile logistics) for comparison:

- **Time-Greedy**, which generates the route by sorting the remaining time until the deadline. We then set a fixed speed for the courier. The time prediction is calculated by dividing the distance between locations by the fixed speed.
- **Distance-Greedy**, which generates the route by sorting the distances step-by-step. We then give the time prediction as the same way of Time-Greedy.
- **OR-Tools** [25], which is a heuristic method that seeks to find the shortest route for the courier.
- **OSquare**, whose core component is machine learning model XGBoost [26]. OSquare outputs the next location at one step, and the whole route is generated recurrently. We then train another XGBoost, to complete the time prediction.
- **DeepRoute** [3], which is a deep model equipped with a Transformer encoder and a attention-based decoder to rank a courier's all unpicked-up packages.
- **FDNET** [1], which utilizes LSTM-based RNN and attention mechanism to predict the driver's route in the food delivery system.
- **Graph2Route** [10], which for the first time, model the locations to be visited as a graph and utilizes a GCN-based encoder and attention mechanism to predict the courier's route.

Note that all the deep learning baselines except FDNET are not proposed for time prediction task in their original paper. Therefore, we develop a time prediction module and plug it right after the route prediction module of those models. Specifically, the plugged time prediction module is a three-layer, fully connected neural network. The time prediction module is trained separately from the original model.

C. Metrics

For route prediction, we introduce HR@k, KRC (Kendall Rank Correlation) and LSD (Location Square Deviation) to evaluate the performance:

- **HR@k**: HR@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:

$$\text{HR@}k = \frac{\hat{\boldsymbol{\pi}}_{[1:k]} \cap \boldsymbol{\pi}_{[1:k]}}{k}. \tag{42}$$

- **KRC**: Kendall Rank Correlation [27] is a statistical criterion to measure the ordinal association between two sequences. We can define a triplet, which be like $(l_i, \hat{o}_i, o_i)$, where l_i is the i-th location to be visited of

TABLE III: Route Prediction Results

Method	$n \in (3-10]$			$n \in (10-20]$			all		
	HR@3	KRC	LSD	HR@3	KRC	LSD	HR@3	KRC	LSD
Distance-Greedy	66.04	0.50	4.71	54.62	0.42	16.72	64.08	0.49	6.78
Time-Greedy	65.48	0.52	4.41	40.02	0.39	18.82	61.09	0.50	6.80
OR-Tools	64.23	0.47	5.21	52.38	0.33	21.01	62.18	0.44	7.93
OSquare	70.33 ±2.28	0.60 ±0.02	3.83 ±0.83	55.59 ±1.58	0.51 ±0.01	14.76 ±1.39	68.12 ±1.14	0.57 ±0.01	5.71 ±0.10
DeepRoute	72.47±0.62	0.61±0.01	3.38 ±0.09	56.86±1.54	0.52±0.01	12.14±0.36	70.08±0.68	0.60±0.01	4.86±0.12
FDNET	50.13±0.06	0.29±0.01	7.43±0.05	23.35±0.02	0.14±0.03	28.83±0.14	35.08±0.02	0.21±0.01	17.81±0.19
Graph2Route	70.53±2.44	0.58±0.03	3.38±0.09	56.86±1.54	0.53±0.01	12.14±0.36	70.08±0.68	0.60±0.01	4.86±0.12
M^{2}G4RTP	**74.46±0.01**	**0.64±0.01**	**3.00±0.05**	**60.38±0.01**	**0.56±0.01**	**11.13±0.25**	**72.21±0.02**	**0.62±0.02**	**4.25±0.08**
Improvement	2.75%	4.92%	11.24%	6.19%	5.66%	8.32%	3.04%	3.33%	12.55%

TABLE IV: Time Prediction Results.

Method	$n \in (3-10]$			$n \in (10-20]$			all		
	RMSE	MAE	acc@20	RMSE	MAE	acc@20	RMSE	MAE	acc@20
Distance-Greedy	84.37	50.11	45.11	78.92	50.00	36.42	82.80	50.08	42.55
Time-Greedy	75.53	44.69	46.46	71.87	46.78	37.29	74.47	45.31	43.75
OR-Tools	82.25	49.03	44.71	76.94	48.72	36.83	80.72	48.94	42.38
OSquare	57.65 ±1.31	39.64 ±1.03	36.63 ±1.59	53.17 ±1.02	36.68 ±0.81	39.46 ±1.18	56.38 ±1.23	38.37 ±0.93	37.47 ±1.47
DeepRoute	40.51±0.02	26.45±0.10	56.22±0.43	42.33±0.20	27.26±0.18	54.22±0.42	40.94±0.07	26.70±0.02	55.54±0.21
FDNET	62.07±5.23	45.86±0.01	28.00±0.01	68.08±0.30	43.86±0.10	32.90±0.02	66.36±013	45.41±0.03	29.29±0.01
Graph2Route	40.51±0.04	26.42±0.06	56.20±0.32	42.35±0.03	27.21±0.06	54.47±0.35	40.96±0.04	26.67±0.06	55.57±0.31
M^{2}G4RTP	**38.98±0.47**	**25.22±0.75**	**59.30±0.68**	**40.10±0.47**	**25.93±0.54**	**56.34±0.66**	**39.30±0.43**	**25.44±0.57**	**58.62±0.26**
Improvement	3.83%	4.62%	5.48%	5.24%	4.74%	3.43%	4.01%	4.61%	5.49%

courier, $\hat{o}_i$ is the order of l_i in the prediction route, and y_i is the order of l_i in the real route. Any pair of $(l_i, \hat{o}_i, o_i)$ and $(l_j, \hat{o}_j, o_j)$ is said to be concordant, if both $\hat{o}_i > \hat{o}_j$ and $o_i > o_j$ or both $\hat{o}_i < \hat{o}_j$ and $o_i < o_j$. Otherwise, it is said to be a discordant pair. KRC is defined as:

$$\mathrm{KRC} = \frac{N_c - N_d}{N_c + N_d}, \tag{43}$$

where N_c is the number of concordant pairs, and N_d is the number of discordant pairs.

- **LSD**: The Location Square Deviation measures the degree that the prediction deviates from the label:

$$\mathrm{LSD} = \frac{1}{N} \sum_{i=1}^{N} \left(\hat{o}_i - o_i\right)^2. \tag{44}$$

For time prediction, we apply Root Mean Square Error (RMSE), Mean Absolute Error (MAE) and Accuracy within 20 minutes (acc@20) as the evaluation metrics, which are formulated as follows:

$$\begin{aligned} \mathrm{RMSE} &= \sqrt{\frac{1}{N} \sum_{i=1}^{N} \left(\hat{y}_i - y_i\right)^2} \\ \mathrm{MAE} &= \frac{1}{N} \sum_{i=1}^{N} \left|\hat{y}_i - y_i\right| \\ \mathrm{acc@20} &= \frac{\sum_{|\hat{y}-y|<20}}{N} \times 100\%, \end{aligned} \tag{45}$$

where $\sum_{|y-\hat{y}|<20}$ denotes the number of samples with an absolute error of time prediction within 20. Because the arrival time varies a lot between locations located at different positions on the route, the traditional percentage error is deceptive. Therefore, this metric is introduced to evaluate the extent to which the model provides acceptable prediction results.

D. Results

The number of locations to be visited by a courier varies from 3 to 20. The difficulty of both route and time prediction increases with the increase of the number of locations. In order to measure the prediction ability of different models under different data scales, we respectively evaluated the performance of the models in the case of 3-10 locations and 11-20 locations. The results are shown in Table III and Table IV, which show that our M^{2}G4RTP model achieves promising performance improvements compared with all the baselines.

For route prediction, all the two greedy-based methods and OR-Tools only take the distance/time into account, whose results are only the optimal solution under a specific strategy, ignoring a large number of spatial-temporal constraints. Therefore they failed to achieve good results. Tree-based model OSquare lacks the ability to model spatial-temporal correlation. Besides, the goal of OSquare is to maximize the output probability of the next one location instead of the entire route. Sequence-based models DeepRoute and FDNET are difficult to model the neighborhood relationship between locations, so they may provide unreasonable outputs. Graph2Route try to solve this problem by introducing a graph-based encoder, but the high-level transfer mode of couriers between AOIs has not been considered. Therefore, the effect of Graph2Route is still weaker than our model. In addition, FDNET is designed for route prediction in the food delivery scenario, where the

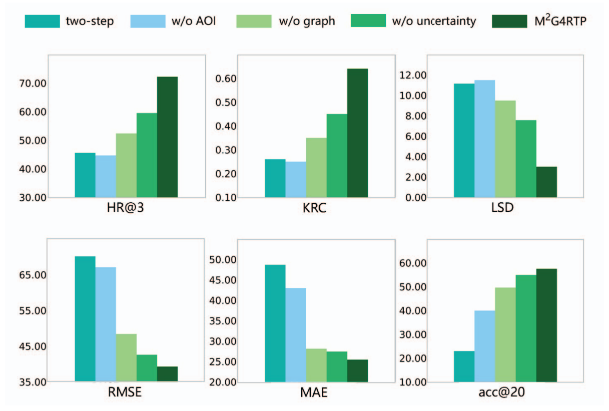


Fig. 5: Component Analysis.

number of locations to be visited is much smaller than that in the logistics scenario. The RNN-based encoder used in FDNET will aggravate the accumulation of errors, which leads to poor performance of FDNET. Our M^{2}G4RTP improves HR@3, KRC, and LSD by 2.75%, 4.92%, and 11.24%, respectively, for short route (3-10) prediction. The improvement of HR@3 and KRC can reach 6.19% and 5.66% in the case of long route (11-20) prediction. This shows that our model has stronger prediction ability in complex scenarios, thanks to the guidance information from AOI-level and the additional monitoring information provided by the multi-tasking learning.

For time prediction, Greedy-base methods, OR-Tools, OSquare, DeepRoute, and Graph2Route do not have the ability to give time prediction originally. The additional plugged time prediction module is trained separately from the original sequence prediction module, and the two modules cannot share the location representation. Therefore, these models failed to achieve satisfactory time prediction results. Due to the disadvantage of the sequence-based encoder and error accumulation caused by the two-stage paradigm, FDNET also failed to achieve promising results in time prediction. In the case of the short sample, our M^{2}G4RTP makes 3.83%, 4.62%, and 5.48% improvement on RMSE, MAE, and acc@20, respectively. Similarly, our model has made greater improvement in the case of long samples for 5.24% and 4.47% on RMSE and MAE, respectively. When the courier has more than 10 locations to visit at the same time, the arrival time of some locations is usually more than two hours. Therefore, a small improvement on acc@20 is still reasonable. Good time prediction results benefit from the ability to avoid error accumulation, which comes from the unique structure of SortLSTM and the multi-task training mode of the model.

E. Component Analysis

In order to study the effectiveness of each model component, we further carried out ablation study. Specifically, we removed some components from the original model and observed the changes in the experimental results. The following three variants are designed in the experiments:

- **two-step**. First, we convert the multi-task training mode of the model into two-stage training, that is, assign an optimizer to the parameters of SortLSTM separately. It is designed to test whether the multi-tasking framework can learn more generalized representation and also avoid the problem of error accumulation.
- **w/o AOI**. Second, we remove all AOI-level information, making the multi-level graph degenerate into a single-level graph of locations. No guidance information from AOI will be introduced in the decoding process of locations. This is to prove that AOI-level information is of great guiding significance.
- **w/o graph**. Third, we use bidirectional LSTM to replace GAT-e as the encoder. This is to prove that the graph-based encoder has a stronger ability to capture the spatial-temporal correlation between locations or AOIs than the sequence-based encoder.
- **w/o uncertainty**. Last, we remove the weight assignment method based on homoscedastic uncertainty. Instead, we manually set the loss weight for the route and time prediction task to 100:1.

Figure 5 shows the results. As we can see, each well-designed module in the M^{2}G4RTP has played an important role, especially the multi-level graph and multi-task training framework. First, the two-step version does not perform well in both route and time prediction. This proves that it is difficult for the encoder to learn effective and generalized representations when there is only single supervision information. Not surprisingly, compared with the original model, the performance of the two-step model in route prediction task degrades more significantly, which is suffering from error accumulation. Second, without courier's high-level transfer mode learned by multi-level graph, it is also difficult for the model to achieve better results. In particular, the route prediction result deteriorates more significantly, which proves the route prediction task especially benefits from the AOI-level information. Third, the results of the model without graph prove that it is necessary to use graph-based encoders to model locations/AOIs because such encoders can better learn the spatial-temporal correlation between nodes. Finally, model without homoscedastic uncertainty weighting is inferior since the predetermined task weight is difficult to meet the needs of heterogeneous multi-task training. The dynamic weight allocation based on homoscedastic uncertainty makes the training process of each task more balanced. Consequently, we get better Pareto boundary [28], namely, better both route and time prediction results.

F. Scalability Analysis

As shown in the table V, we analyze the time complexity of the models, where F refers to the number of features, N and A refer to the number of locations and AOIs contained in a sample, E refers to the number of edges in location graph $\boldsymbol{G^l}$, t and d refer to the number and the height of trees in XGBoost. Besides, the time (in ms) required for each model to make an inference under the optimal parameters is compared.

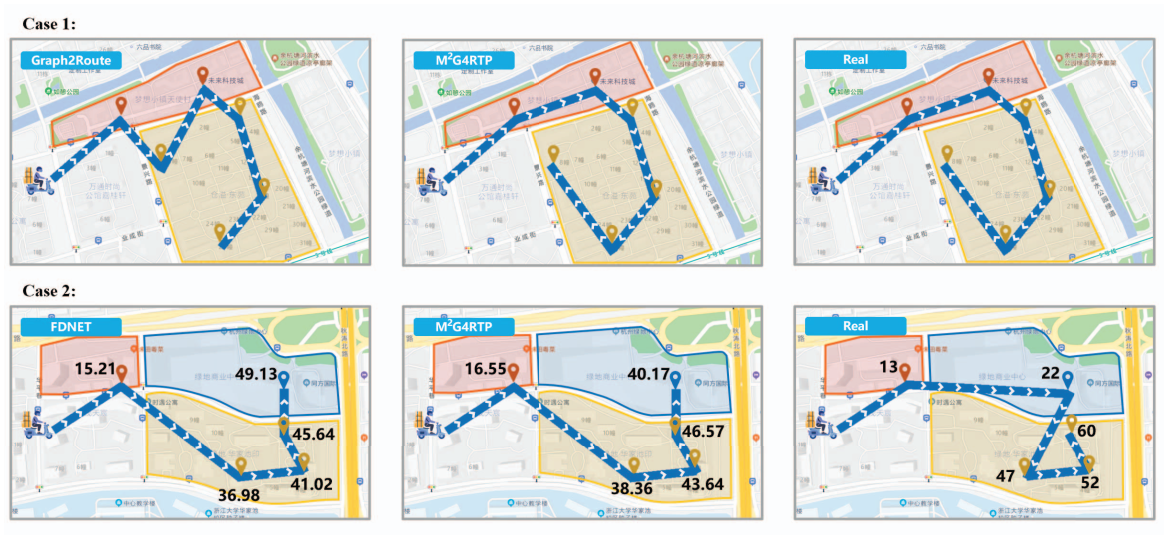


Fig. 6: Case Study. The blue lines are real route or routes predicted by different models.

TABLE V: Scalability Analysis.

Method	Inference Time Complexity	Inference Time
Greedy-based	$O(N \log N)$	0.12
OSquare	$O(tdFN)$	0.39
DeepRoute	$O(N^2F + NF^2 + N^2F^2)$	0.34
Graph2Route	$O(NF^2 + EF^2 + N^2F^2)$	0.39
FDNET	$O(NF^2 + N^2F^2)$	0.31
M²G4RTP	$O(NF^2 + EF^2 + N^2F^2 + A^2F^2)$	0.61

The multi-level graph encoder in M^{2}G4RTP can run in parallel, and $A \ll N$, so the time consumption of AOI-level GAT-e can be ignored. Since the location-level route and time prediction require the AOI-level prediction results as guidance information, the multi-task decoder operates in a serial manner with time complexity of $O(N^2F^2 + A^2F^2)$. Since M^{2}G4RTP models the transfer mode of couriers between AOIs, it inevitably introduces additional time complexity compared with other models. However, the prediction of route and time only occurs when the set of locations to be visited changes. Therefore, for a single courier, the interval between requests for prediction service may be as high as tens of minutes. Considering that the performance of M^{2}G4RTP is about 3%-12% higher than the existing baselines, we regard this is a pretty good deal between efficiency and effectiveness.

G. Case Study

To analyze the performance of M^{2}G4RTP more intuitively, We provide an empirical case study shown in Figure 6.

In the first case, Graph2Route outputs a route that travels between two communities multiple times, which is obviously unreasonable. Although this route seems to have a shorter total distance, in the actual scenario, it is difficult for couriers to freely and quickly shuttle between the two communities. On the contrary, it is more common to visit another community after traversing all the locations of one community. Graph2Route lacks the ability to model the high-level transfer mode of couriers between AOIs, which makes it difficult to make accurate predictions in such complex situations.

In the second case, neither M^{2}G4RTP nor FDNET can predict the route that is exactly consistent with the real M^{2}G4RTP. However, thanks to the tolerance of its multi-task framework for errors between different tasks and the correction ability of SortLSTM to overcome the incorrect route information, M^{2}G4RTP gives more acceptable time prediction results. RMSE of the entire sample is FDNET 15.28 and M^{2}G4RTP 11.56, and MAE of the entire sample is FDNET 12.94 and M^{2}G4RTP 10.43, respectively.

VI. Deployment and Applications

A package pick-up route and time inference system based on M^{2}G4RTP is deployed in Cainiao, which supports the main needs of both the user and the courier. The system has been put into an actual environment with hundreds of thousands of queries per day.

A. System Deployment

The system is shown in Figure 7, which consists of Feature Extraction Layer (Offline), Inference Layer (Online), and Application Layer (Online). Given an RTP request, the Feature Extraction Layer resolves the courier's current location and all locations and AOIs to be visited. Distance features were calculated from the distance tool in Graph Builder, and multi-level graph is constructed based on the affiliation relationship between locations and AOIs. The pre-trained model is packaged as M^{2}G4RTP Service module, which outputs route and time prediction results based on the input multi-level graph. Currently, the applications deployed at the Application Layer include Intelligent Order Sorting Service and Minute-Level ETA Service for couriers and users, respectively.

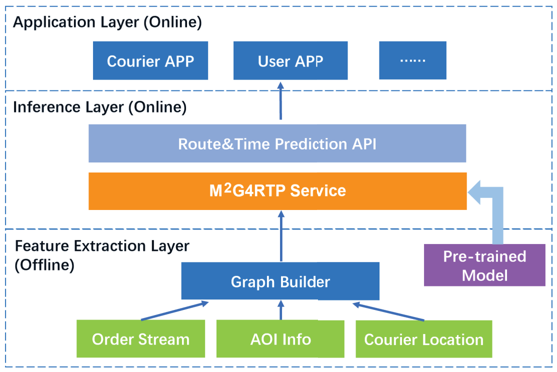
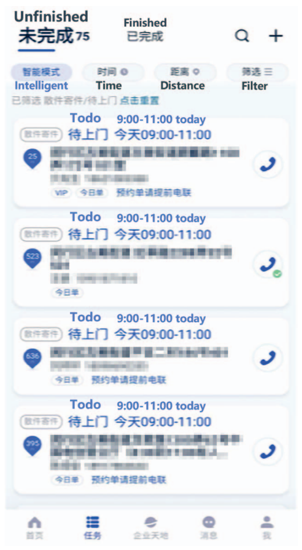
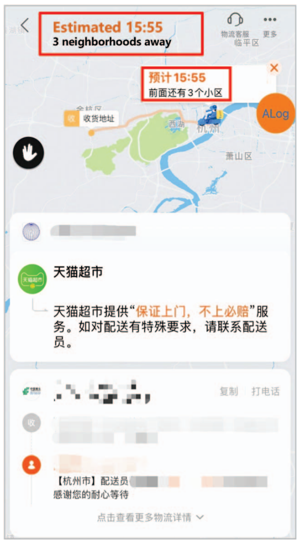


Fig. 7: The deployment details of M^{2}G4RTP in online environment.



(a) Intelligent Order Sorting Service



(b) Minute-level ETA Service

Fig. 8: Applications on Cainiao APP.

B. Intelligent Order Sorting Service

The intelligent order sorting service is deployed to serve couriers in the package pick-up scenario. The route prediction results can greatly facilitate the work of the courier, for it provides a more intelligent way of sorting orders, that is, ranking orders according to the possible future route of the courier, which is shown in Figure 8(a). Previously, the platform could only show all the courier's unpicked-up orders in a time-greedy or distance-greedy way, which made the courier need to read all the orders to plan his own route. After intelligent sorting based on route prediction results, the order list will be more in line with the couriers' working habits, reducing the burden for the couriers. According to the feedback from the couriers in Shanghai, the Intelligent Order Sorting Service basically meets their expectations and can effectively improve the efficiency of picking-up packages. The HR@3 and KRC of route prediction can reach 66.89% and 0.61.

C. Minute-level ETA Service

Different from package or food delivery service, package pick-up is a face-to-face service that requires customers to be present during service. Users need to keep themselves available until the courier arrives. Therefore, users have more waiting anxiety, and the demand for accurate ETA (estimated time of arrival) is much stronger. The route and time prediction results provide users with more reliable ETA services. Previously, the ETA service that the platform provided to users specifically referred to a period of 2 hours. Right now, precise route and time prediction have the ability to provide users with minute-level ETA services and inform them how far the couriers are, shown in Figure 8(b). On the basis of the accurate ETA service, we send a push or text message to the user before the courier arrives so that the user can be prepared in advance and reduce their waiting anxiety. After the deployment, customers' complaints because of inaccurate predictions decreased significantly. According to the data collected from Shanghai, the RMSE and MAE of ETA can be as low as 31.11 and 22.40.

VII. Conclusion

In this paper, we try to solve the problem of RTP in the field of instant-logistics. For the first time, we model multi-level transfer mode of couriers between locations and AOIs and simultaneously predicted the route and time in a multi-task manner. We propose GAT-e to model the spatial-temporal correlation between nodes in multi-level graph, which overcomes the limitations of sequence-based model. A multi-task decoder equipped with a mask attention mechanism and SortLSTM is designed to simultaneously give route and time prediction. Finally, extensive experiments carried out on an industry-scale real-world dataset, as well as an online deployment system, demonstrate the superiority of our proposed model.

Acknowledgment

This work was sponsored by CCF-Alibaba Innovative Research Fund For Young Scholars. The experimental data and deployment environment are provided by Cainiao Network, Alibaba.

References

[1] 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.

[2] F. Wu and L. Wu, "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*, 2019, pp. 774–781.

[3] 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.

[4] Y. Zhang, Y. Liu, G. Li, Y. Ding, N. Chen, H. Zhang, T. He, and D. Zhang, "Route prediction for instant delivery," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 3, no. 3, p. Article 124, 2019.

[5] A. C. de Araujo and A. Etemad, "End-to-end prediction of parcel delivery time with deep learning for smart-city applications," *IEEE Internet of Things Journal*, 2021.

[6] Z. Wang, K. Fu, and J. Ye, “Learning to estimate the travel time,” in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, Y. Guo and F. Farooq, Eds. ACM, 2018, pp. 858–866.

[7] Y.-L. Lan, F. Liu, W. W. Ng, J. Zhang, and M. Gui, “Decomposition based multi-objective variable neighborhood descent algorithm for logistics dispatching,” *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2020.

[8] J. Wang, M. K. Lim, Y. Zhan, and X. Wang, “An intelligent logistics service system for enhancing dispatching operations in an iot environment,” *Transportation Research Part E: Logistics and Transportation Review*, vol. 135, p. 101886, 2020.

[9] W. Huang, Z. Zhao, X. An, G. Min, and J. Li, “Dynamic scheduling for urban instant delivery with strict deadlines,” in *ICC*, 2020, pp. 1–6.

[10] 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 *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 4143–4152.

[11] Y. Tong, Y. Zeng, Z. Zhou, L. Chen, J. Ye, and K. Xu, “A unified approach to route planning for shared mobility,” *Proceedings of the VLDB Endowment*, vol. 11, no. 11, p. 1633, 2018.

[12] Y. Zeng, Y. Tong, and L. Chen, “Last-mile delivery made practical: An efficient route planning framework with theoretical guarantees,” *Proceedings of the VLDB Endowment*, vol. 13, no. 3, pp. 320–333, 2019.

[13] Y. Tong, Y. Zeng, Z. Zhou, L. Chen, and K. Xu, “Unified route planning for shared mobility: An insertion-based framework,” *ACM Transactions on Database Systems (TODS)*, vol. 47, no. 1, pp. 1–48, 2022.

[14] I. Jindal, X. Chen, M. Nokleby, J. Ye *et al.*, “A unified neural network approach for estimating travel time and distance for a taxi trip,” *arXiv preprint arXiv:1710.04350*, 2017.

[15] Y. Li, K. Fu, Z. Wang, C. Shahabi, J. Ye, and Y. Liu, “Multi-task representation learning for travel time estimation,” in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2018, pp. 1695–1704.

[16] J. Hu, B. Yang, C. Guo, C. S. Jensen, and H. Xiong, “Stochastic origin-destination matrix forecasting using dual-stage graph convolutional, recurrent neural networks,” in *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 2020, pp. 1417–1428.

[17] D. Wang, J. Zhang, W. Cao, J. Li, and Y. Zheng, “When will you arrive? estimating travel time based on deep neural networks,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[18] Z. Wang, K. Fu, and J. Ye, “Learning to estimate the travel time,” in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 858–866.

[19] J. Qiu, L. Du, D. Zhang, S. Su, and Z. Tian, “Nei-tte: intelligent traffic time estimation based on fine-grained time derivation of road segments for smart city,” *IEEE Transactions on Industrial Informatics*, vol. 16, no. 4, pp. 2659–2666, 2019.

[20] K. Fu, F. Meng, J. Ye, and Z. Wang, “Compacteta: A fast inference system for travel time prediction,” in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 3337–3345.

[21] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, “Graph attention networks,” *arXiv preprint arXiv:1710.10903*, 2017.

[22] C. A. Hoare, “Quicksort,” *The computer journal*, vol. 5, no. 1, pp. 10–16, 1962.

[23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” *Advances in neural information processing systems*, vol. 30, 2017.

[24] A. Kendall, Y. Gal, and R. Cipolla, “Multi-task learning using uncertainty to weigh losses for scene geometry and semantics,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7482–7491.

[25] Google. Or-tools, google optimization tools. (2016). [Online]. Available: https://developers.google.com/optimization/routing

[26] T. Chen and C. Guestrin, “Xgboost: A scalable tree boosting system,” in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

[27] M. G. Kendall, “A new measure of rank correlation,” *Biometrika*, vol. 30, no. 1/2, pp. 81–93, 1938.

[28] Y. Sawaragi, H. NAKAYAMA, and T. TANINO, *Theory of multiobjective optimization*. Elsevier, 1985.