RESEARCH-ARTICLE

Congestion-aware Spatio-Temporal Graph Convolutional Network-based A* Search Algorithm for Fastest Route Search

HONGJIE SUI, Tsinghua University, Beijing, China

HUAN YAN, Tsinghua University, Beijing, China

TIANYI ZHENG, Tsinghua University, Beijing, China

WENZHEN HUANG, Tsinghua University, Beijing, China

YUNLIN ZHUANG, Hitachi China Ltd., Hong Kong, China

YONG LI, Tsinghua University, Beijing, China

Open Access Support provided by:

Tsinghua University

Hitachi China Ltd.

Published: 19 June 2024
Online AM: 11 April 2024
Accepted: 31 March 2024
Revised: 30 December 2023
Received: 12 March 2023

Congestion-aware Spatio-Temporal Graph Convolutional Network-based A* Search Algorithm for Fastest Route Search

HONGJIE SUI, Department of Electronic Engineering, Tsinghua University, Beijing, China
HUAN YAN, Department of Electronic Engineering, Tsinghua University, Beijing, China
TIANYI ZHENG, Department of Electronic Engineering, Tsinghua University, Beijing, China
WENZHEN HUANG, Department of Electronic Engineering, Tsinghua University, Beijing, China
YUNLIN ZHUANG, Hitachi (China) Research & Development Corporation, Beijing, China
YONG LI, Department of Electronic Engineering, Tsinghua University, Beijing, China

The fastest route search, which is to find a path with the shortest travel time when the user initiates a query, has become one of the most important services in many map applications. To enhance the user experience of travel, it is necessary to achieve accurate and real-time route search. However, traffic conditions are changing dynamically, and the frequent occurrence of traffic congestion may greatly increase travel time. Thus, it is challenging to achieve the above goal. To deal with it, we present a congestion-aware spatio-temporal graph convolutional network-based A* search algorithm for the task of fastest route search. We first identify a sequence of consecutive congested traffic conditions as a traffic congestion event. Then, we propose a spatio-temporal graph convolutional network that jointly models the congestion events and changing travel time to capture their complex spatio-temporal correlations, which can predict the future travel-time information of each road segment as the basis of route planning. Further, we design a path-aided neural network to achieve effective origin-destination (OD) shortest travel-time estimation by encoding the complex relationships between OD pairs and their corresponding fastest paths. Finally, the cost function in the A* algorithm is set by fusing the output results of the two components, which is used to guide the route search. Our experimental results on the two real-world datasets show the superior performance of the proposed method.

CCS Concepts: • **Information systems → Spatial-temporal systems;**

Additional Key Words and Phrases: A* search algorithm, spatio-temporal mining, traffic congestion, travel-time estimation

This work was supported in part by The National Key Research and Development Program of China under grant 2020YFA0711403, the National Nature Science Foundation of China under 62272260, U20B2060.

Authors' addresses: H. Sui, H. Yan (Corresponding author), T. Zheng, W. Huang, and Y. Li, Department of Electronic Engineering, Tsinghua University, Beijing 100084, China; e-mails: huihj21@mails.tsinghua.edu.cn, yanhuanthu@gmail.com, zhengzong20010904@163.com, huangwenzhen@tsinghua.edu.cn, liyong07@tsinghua.edu.cn; Y. Zhuang, Hitachi (China) Research & Development Corporation, Beijing 100190, China; e-mail: ylzhuang@hitachi.cn.

ACM 1556-4681/2024/06-ART179
https://doi.org/10.1145/3657640

ACM Reference Format:
Hongjie Sui, Huan Yan, Tianyi Zheng, Wenzhen Huang, Yunlin Zhuang, and Yong Li. 2024. Congestion-aware Spatio-Temporal Graph Convolutional Network-based A* Search Algorithm for Fastest Route Search. *ACM Trans. Knowl. Discov. Data.* 18, 7, Article 179 (June 2024), 19 pages. https://doi.org/10.1145/3657640

1 INTRODUCTION

Online map applications like Google Maps that provide users with valuable and real-time navigation information have brought great convenience for people's daily travel. The fastest route search, as one of the most important functions embedded in them, would greatly reduce the user travel cost and enhance the experience of travel. Its objective is to find a route with the shortest travel time after receiving a user query that includes a start point, a destination, and a departure time, as shown in Figure 1. Due to the complex and time-varying traffic conditions, it is challenging to achieve accurate fastest route searching.

Many existing works focus on extending the heuristic search algorithm like Dijkstra and A* search algorithm on a time-dependent graph constructed from the road network and the corresponding traffic data [5, 25, 37]. By elaborately designing the heuristics, a high-quality solution for our studied task can be obtained. Thus, it is critical to learn a suitable cost function for heuristic search algorithms. A* algorithm is a classical heuristic search algorithm whose goal is to generate a route from a start point to the destination with minimal cost. The main idea of this algorithm is to determine which path to extend based on a cost function, which consists of the observable cost function and estimated cost function. The observable cost function is defined as the total cost of the path from the start point to the candidate point, and the estimated cost is a heuristic function that estimates the cost from the candidate point to the destination. Early works use statistical methods to develop the heuristic function, resulting in low flexibility and accuracy [5, 17, 27]. The main disadvantage of the statistical methods is that they are insufficient for capturing the complex patterns of dynamic traffic conditions. With the advances in deep learning techniques, some recent studies design deep neural networks for the fastest route search task [31, 39]. For example, authors in Reference [39] propose to extend the A* search algorithm by using neural networks to model the dynamic traffic patterns. Based on this work, Wang et al. [31] adopt an adaptive graph convolutional recurrent network and a multi-task representation learning approach to improve the performance of the search algorithm. However, the above methods still have two following limitations:

— *The complexity of the influence of traffic congestion.* Traffic congestion usually occurs on road segments, which greatly influences the corresponding travel time. For example, the road segment connecting the residential area and business area could be congested during rush hours, resulting in long travel time. On the one hand, the occurrence of traffic congestion exhibits obvious time periodicity. On the other hand, traffic congestion sparsely distributes at different time, e.g., the time intervals between congestion events may be several minutes or even hours. Recognizing these patterns within the dynamic nature of traffic flows presents a challenge, and thus how to explicitly model traffic congestion is a challenge.

— *Unknown path information for travel-time estimation.* How to effectively set the estimated cost function determines the final performance of the A* search algorithm. Due to limited query information, we do not have the path information for the shortest travel-time estimation, which makes it challenging to achieve an accurate estimation. Some works adopt either top K shortest paths [39] or multi-task representation learning approach [31] to address it. However, these methods still have drawbacks. To be specific, in Reference [39], estimating

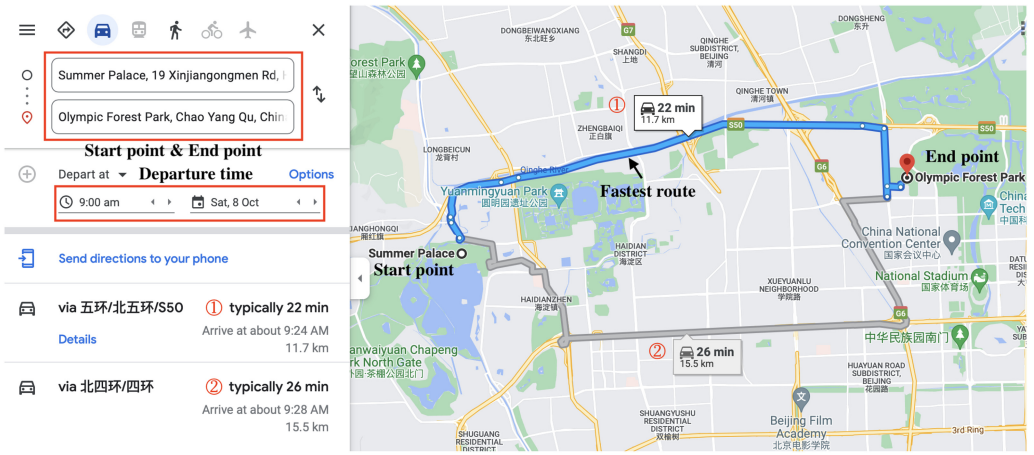


Fig. 1. An example of fastest route search in Google map.

the cost at each expansion of the A* algorithm by performing the shortest path search can be a time-consuming process, and it is possible that the fastest path may not be among the top K shortest routes. The multi-task learning method proposed in Reference [39] does not incorporate any path information between the start and end points, which would greatly influence the estimation accuracy. Thus, these methods do not adequately address this challenge.

To overcome the aforementioned limitations, we propose a congestion-aware spatio-temporal graph convolutional network based on the A* search algorithm to solve the problem of finding the fastest path. First, we identify a sequence of consecutive congested traffic conditions as a traffic congestion event. Then, we design a congestion-aware deep learning module based on the graph convolutional network and temporal convolutional network to predict the travel time of road segments. This module is able to capture the complex spatio-temporal traffic condition by explicitly modeling the traffic congestion events. Further, we build a path-aided **origin-destination (OD)** travel-time estimation model that encodes the complex relationship between OD pairs and their corresponding fastest paths to estimate the shortest travel time. Then, the observable cost function is computed by adding the observed travel time with the predicted travel time of a candidate road segment, and the estimated cost is set based on the estimated shortest travel time. Finally, we perform the normal search process of the A* algorithm by integrating these two cost functions.

Overall, we summarize our main contributions as follows:

— We design a congestion-aware neural network that explicitly models the influence of traffic congestion on travel-time estimation.
— We propose a path-aided neural network for OD shortest travel-time estimation, which binds the OD pairs with their corresponding fastest paths by bringing their hidden representations closer during the training phase. With the useful information learned from the affiliated paths, we only use the OD input and departure time to estimate the shortest travel time at the estimation phase.
— Extensive experiments are conducted in two real-world datasets, and the results demonstrate the performance superiority of our proposed model.

We describe the remainder of this article as follows. First, we review the literature on fast route search in Section 2. Then, we give the key definitions and present our research problem in Section 3.

Next, we describe our proposed approach in Section 4. Further, we conduct the various experiments and analyze the experiment results in Section 5. Finally, we conclude our work in Section 6.

2 RELATED WORK

In this section, we will introduce several important works about our research, which fall into the following three major groups.

Road Segment Travel Time Estimation. The task of road segment travel-time estimation is to infer the travel time on individual road segments, which attracts much attention from the industry and academia [4, 13, 28, 29, 35, 38]. For instance, the authors of Reference [13] proposed to use a spatial moving average structure to capture the correlations between travel time on different road segments. Reference [12] proposes four dynamic graph miners for travel cost extraction, which can be selected and combined to extract travel costs that cater to different user requirements. With the advance of deep learning techniques in traffic [2, 10, 18, 20, 34, 42], more researchers focus on designing various neural networks to improve the accuracy of travel-time estimation [14, 32, 43]. For example, Google employed a graph neural network to estimate the travel time of roads in its practical systems [4]. Jin et al. [14] adopted a hierarchical neural architecture search approach to automatically learn a high-quality neural network for travel-time estimation. The road segment travel-time estimation is a critical function in our work, and its difference lies in capturing the complex spatial and temporal patterns of traffic conditions by explicitly modeling the congestion features.

Fastest Route Search. There are many works studying the task of fastest route search based on the traffic condition information, aiming to find the path with the shortest travel time [5, 9, 17, 21, 26, 39, 46–48]. For example, Ding et al. [5] introduced a novel algorithm to find time-dependent shortest paths over a road network. The authors of Reference [17] extended the A* algorithm to find the fastest paths. Yuan et al. [47, 48] utilized the historical vehicle trajectories to estimate the distribution of travel time of each road segment in different periods of time and then adopted a two-step routing algorithm to calculate the fastest route. Recently, Wu et al. [39] proposed to leverage the neural networks to model complex traffic information based on the A* algorithm, which can achieve higher accuracy than other state-of-the-art baselines. In contrast, our work proposes a congestion-aware deep learning model to fully capture the spatial and temporal patterns of traffic conditions. Meanwhile, a deep learning method is developed to accurately estimate the travel time based on the given OD information.

Deep Learning for Heuristic Search. The heuristic function plays a vital role in aiding the A* algorithm's search by estimating the travel time from the candidate node to the destination node. Existing deep learning-based methods for heuristic value estimation can be grouped into two different categories. The first category is based on path search [6, 30, 36, 39, 41, 44]. NASF [39] employs Dijkstra's algorithm to explore multiple paths and predict the travel time of each path using neural networks. However, the high computational complexity of this approach makes it difficult to apply to practical route planning. The second one is based on OD travel-time estimation [16, 22, 31, 33], which aims to predict travel time without determining an actual route. DeepOD [45] encodes the trajectory information into the hidden representation of the corresponding OD pair during the model training phase. During the testing phase, DeepOD predict the travel time only with the OD input. However, this model does not effectively capture the complex spatio-temporal dependencies of the road network. Meanwhile, since there is no information on fast paths available, it cannot directly be used to estimate the shortest travel time for an OD query. To address these limitations, on the one hand, we leverage historical road condition information and encode it for each road segment. Moreover, we use the time-dependent Dijkstra algorithm to generate a large number of the

fastest paths based on historical data, which provides more valuable information for the shortest travel-time estimation.

3 PRELIMINARIES

In this section, we give the key definitions and present our research problem. Further, we describe the A* search algorithm that is widely used in solving the path-finding problem.

3.1 Definitions

Road Network. Consider a directed graph $G = (V, E)$ to represent a road network, where $V = \{r_1, r_2, ..., r_N\}$ is a set of N nodes and E is a set of edges. Each node denotes a road segment. An edge $e =< r_i, r_j > \in E$ denotes the directed connection from road segment r_i to road segment r_j.

Travel Time of Road Segments. For road segment k, its travel time l_t^k is defined as the average travel time of vehicles during the time slot t. Typically, the travel time of a road segment changes at different time slots under the time-varying traffic condition, e.g., the travel time is long in rush hours but could be short at other periods of time.

Traffic Condition Level: We classify the traffic condition into three levels: unobstructed, slow and congested. Such classification is specified according to the criteria used by Baidu Map as an example. It is assumed that the traffic condition on a road segment has a unique level at a specific time slot.

Traffic Congestion Event: We identify a traffic congestion event based on the traffic condition levels. A congestion event c_i is denoted as a sequence of the specified traffic condition levels on road segment i, which is expressed as $[(t_s, s_{i,t_s}), (t_s + 1, s_{i,t_s+1}) ..., (t_e, s_{i,t_e})], t_s < t_e$. It satisfies the following two conditions:

(1) The value of $s_{i,l} (t_s \leq l \leq t_e)$ corresponds to slow or congested level.
(2) The values of s_{i,t_s-1} and s_{i,t_e+1} are unobstructed level.

Route. A route y is a sequence of connected road segments, i.e., $y : r_1 \rightarrow r_2 \rightarrow ... \rightarrow r_n$. The start and destination points of route y are denoted as $y.s = r_1.s$ and $y.d = r_n.d$.

3.2 Problem Formulation

Fastest Route Search. Given a query with a start point l_s, an endpoint l_e, and a departure time t_d, we aim to find the fastest route y based on a dataset D consisting of historical traffic information (i.e., the travel time of road segments) in a road network G,

$$y, y.t \leftarrow f(l_s, l_e, t_d | D, G), \tag{1}$$

where $y.t$ denotes the travel time of route y.

3.3 A* Search Algorithm

A* search algorithm [11] is a graph search algorithm, which aims to find a path from the source node n_s to the destination node n_d with the objective of the minimum cost. To implement it, it maintains a path tree, originating from the source node and iteratively selects a path for extension. The algorithm terminates when the desired path is searched or there are no paths to extend. At each path extension, the cost of a candidate node n is calculated by a cost function $f(n)$,

$$f(n) = g(n) + h(n), \tag{2}$$

where $g(n)$ represents the observable cost, which is the cost of the observable path from the source node n_s to the candidate node n. $h(n)$ is the heuristic cost, which estimates the cost of the fastest path from the candidate node n to the destination node n_d.

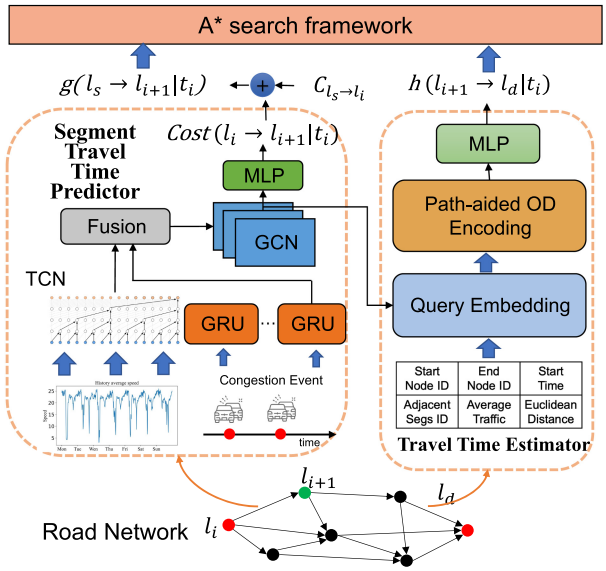


Fig. 2. An overview of our model.

4 OUR APPROACH

In this section, we describe a novel A* search model that incorporates a congestion-aware travel-time prediction module and a path-aided OD travel-time estimation module and explain how this model can solve the two limitations we are concerned about.

4.1 Model Overview

The overall architecture of our model is shown in Figure 2. The model is built upon the framework of the A* search algorithm, so the cost function can be decomposed into the observable cost $g(\cdot)$ and the heuristic cost $h(\cdot)$.

For $g(\cdot)$, it is calculated by summing over the travel time of all the observable road segments. Suppose there is a partial path denoted by $p : r_s \rightarrow r_1 \cdots \rightarrow r_i$, the observable cost of a candidate location r_{i+1} can be computed as follows:

$$\begin{aligned} g(r_s \rightarrow r_{i+1}) &= \sum_{k=1}^{i} Time(r_k \rightarrow r_{k+1}|t_k, q) \\ &= C_{r_s \rightarrow r_i} + Time(r_i \rightarrow r_{i+1}|t_i, q), \end{aligned} \tag{3}$$

where $C_{r_s \rightarrow r_i}$ denotes the observable travel time from r_s to r_i. The second term represents the predicted travel time from r_i to r_{i+1} at departure t_i. To compute it, we design a congestion-aware spatio-temporal prediction model to infer the travel time of each road segment according to the time-varying road condition information in the road network. In particular, we model the influence of traffic congestion events on the travel time of each road segment, which will be described in Section 4.2. Therefore, we define the function $g(r_s \rightarrow r_{i+1})$ as the sum of the predicted travel time and the accumulated travel time up to r_i.

For $h(\cdot)$, we propose a path-aided neural network for OD travel-time estimation. In this network, the query embedding component is used to extract the embeddings of the input features. We employ one-hot encoding to represent the query's origin and destination and then extract the

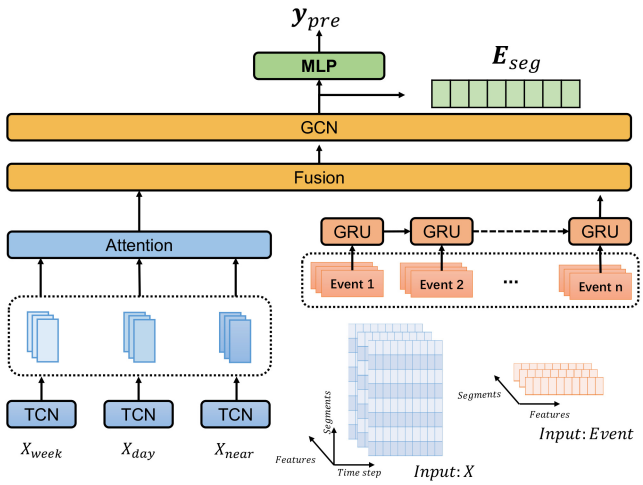


Fig. 3. The architecture of congestion-aware segment travel-time prediction module.

spatiotemporal embeddings of their adjacent road segments based on the results from Section 4.2. Additionally, we calculate the Euclidean distance between the origin and destination nodes, along with the average traffic volume within the road network. These embeddings are concatenated to form the query embedding, as elaborated in Section 4.3. The path-aided OD encoding component is designed to learn the hidden OD representation. Its core is to employ an auxiliary neural network to learn the spatio-temporal representation of the fastest paths between origin and destination and make it close to the hidden representation of the corresponding OD pairs during the training phase. After that, the learned OD representation is input to a **Multi-Layer Perceptron (MLP)** component for travel-time estimation. In the estimation phase, we only input the OD pairs and departure time to generate the estimated shortest travel time.

When the two functions are learned, the travel time of a candidate location can be computed for path extension.

4.2 Congestion-aware Segment Travel Time Prediction

To set observable cost function $g(n)$, we need to estimate the travel time of each road segment. However, since the traffic condition is time varying, especially the traffic congestion results in longer travel time, it is challenging to accurately estimate it. To tackle it, inspired by Reference [19], we propose a congestion-wise spatio-temporal learning model, which explicitly models the congestion information. As shown in Figure 3, this model consists of two important modules. The first is the spatio-temporal feature extraction module, which captures the temporal features of segment travel time. The second is the congestion event information extraction module, which extracts high-density sequential congestion information.

In the spatio-temporal feature extraction module, to better extract the periodic information of road segment travel time, we extract three types of temporal information: an hour before the predicted time period, the predicted time period 1 day ago, and the predicted time period 1 week ago. Each type of temporal information contains the travel-time information of the road segment 1 hour after the corresponding moment, where each 5-minute interval represents a time slice. The corresponding data are denoted by $\mathbf{X}_{near}$, $\mathbf{X}_{day}$, and $\mathbf{X}_{week}$, respectively. To enhance the model's ability to process time-series data, we use a gated one-dimensional **temporal convolutional network (TCN)** [3] to extract temporal features of road segments. The TCN uses dilated causal convolution

as its temporal convolution layer. The dilated causal convolution is represented as

$$\mathbf{x} \star \mathbf{f}(t) = \sum_{s=0}^{K-1} \mathbf{f}(s)\mathbf{x}(t - d \times s), \tag{4}$$

where $\mathbf{x} \in \mathbf{R}^T$ represents a one-dimensional sequence input and $\mathbf{f} \in \mathbf{R}^K$ denotes the filter. t represents the current step, and d is the dilation factor that controls the skipping distance.

In this article, we adopt a three-layer convolutional TCN architecture with a kernel size of 2 and dilation factors of 1, 2, and 4, respectively. Each input is processed through two TCN layers. After applying the sigmoid activation function to the output of the TCN, this output serves as a gating signal that interacts with the preceding output to derive the final time-domain representation. We express it as

$$\mathbf{Y} = \phi(\mathbf{\Theta}_1 * \mathbf{S} + \mathbf{b}_1) \odot \sigma(\mathbf{\Theta}_2 * \mathbf{S} + \mathbf{b}_2), \tag{5}$$

where $\mathbf{S}$ is the input of TCN; $\mathbf{\Theta}_1, \mathbf{\Theta}_2, \mathbf{b}_1, \mathbf{b}_2$ are learnable parameters. The symbols ϕ and σ represent the *tanh* and *sigmoid* activation functions, respectively.

Correspondingly, we input $\mathbf{X}_{week}, \mathbf{X}_{day}, \mathbf{X}_{near}$ into the TCN to get the outputs $\mathbf{Y}_w, \mathbf{Y}_d, \mathbf{Y}_n$. After that, we use the attention mechanism to fuse the outputs of the three channels as the initial features of each road segment, which is formulated as follows:

$$\begin{aligned} att(\mathbf{Y}_1, \mathbf{Y}_2) &= \mathbf{v}^T \tanh(\mathbf{W}_1\mathbf{Y}_1 + \mathbf{W}_2\mathbf{Y}_2 + \mathbf{d}), \\ \alpha_d &= \frac{att(\mathbf{Y}_d, \mathbf{Y}_n)}{att(\mathbf{Y}_d, \mathbf{Y}_n)) + exp(att(\mathbf{Y}_w, \mathbf{Y}_n))}, \\ \alpha_w &= \frac{att(\mathbf{Y}_w, \mathbf{Y}_n)}{att(\mathbf{Y}_d, \mathbf{Y}_n)) + att(\mathbf{Y}_w, \mathbf{Y}_n)}, \\ \mathbf{Y} &= \alpha_d\mathbf{Y}_d + \alpha_w\mathbf{Y}_w, \end{aligned} \tag{6}$$

where $\mathbf{v}, \mathbf{W}_1, \mathbf{W}_2, \mathbf{d}$ are learnable parameters and $\mathbf{Y}$ is the temporal representation of each road segment.

In the congestion event information extraction module, we first summarize the characteristics of traffic congestion, these features include the time embedding, duration, congestion level, and the length of time from the end of the congestion event to the current moment. To encode the time of day and day of the week for congestion events, we apply a one-hot encoding technique. Afterwards, these encoded features are concatenated to generate the time embedding representation. Moreover, we utilize one-hot encoding to encode the congestion level. Finally, these embeddings are concatenated with other relevant features to obtain the final encoding for the congestion event. We select the features of the m recent congestion events at the current node as input. Since the occurrence of congestion has a strict temporal order, we use Gated Recurrent Unit to aggregate the congestion information to obtain the congestion information representation of road segments.

The temporal representations of road segments and the representations of congestion information are aggregated by a multi-layer MLP, denoted by $\mathbf{z}_u^0$. Then we adopt the **graph convolutional network (GCN)** to capture their spatial dependencies, which can be computed as

$$\mathbf{h}_v^{(l+1)} = f\left(\frac{1}{|N(v)|}\sum_{u \in N(v)}\left(\mathbf{W}\mathbf{z}_u^{(l)} + \mathbf{b}^{(l)}\right)\right), l = 0, 1, \ldots, K, \tag{7}$$

where $\mathbf{W}, \mathbf{b}$ are learnable parameters. $N(v)$ represents all the nodes adjacent to node v. In this study, road segments are represented as nodes, and the edges represent the connections between these road segments. It is important to recognize that distance plays a crucial role in determining travel

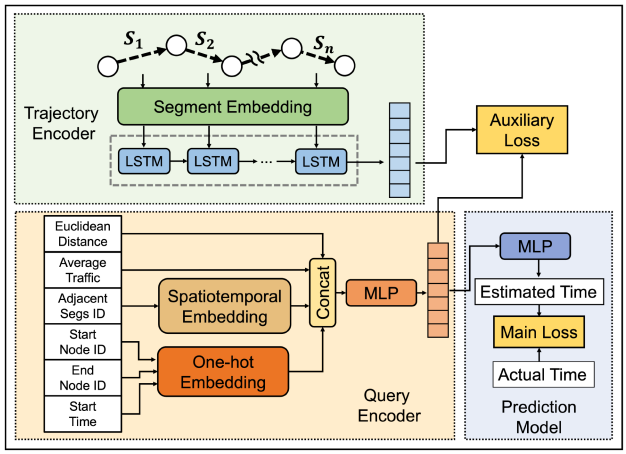


Fig. 4. The architecture of path-aided OD travel-time estimation module.

time. Incorporating this factor into information aggregation process has potential to improve the prediction accuracy.

Finally, the travel time of the road segments is obtained through a fully connected network. We refer to the predicted travel time of road segments as a one-step prediction, which is a part of the observable cost of the current node's child nodes.

4.3 Path-aided OD Travel Time Estimation

The key part in the A* search algorithm is to set the heuristic cost function $h(\cdot)$, which aims to estimate the travel time from the candidate location to the destination. However, since the route from the candidate location to the destination is unknown, it is more difficult to learn $h(\cdot)$ than $g(\cdot)$. Although some recent works [31, 39] attempt to address this challenge, they still cannot effectively exploit the supervised information. As a matter of fact, historical data about travel time allow us to generate the shortest path and estimate travel time for a given OD pair and departure time. Compared with the simple input of the OD pair and the departure time, the affiliated path provides more valuable information about why it takes such a long travel time from origin to destination. Motivated by it, we propose a path-aided neural network for OD shortest travel-time estimation, as shown in Figure 4. Specifically, during the training stage, we encode the corresponding paths to learn the hidden representation of OD pairs. At the estimation stage, we only use the information of the OD pair and the departure time to learn its hidden representation, which is utilized to estimate the travel time.

When encoding the OD pair, we not only consider the current candidate node ID and destination node ID but also extract other contextual traffic information, which includes the following: (1) the current departure time t_i, which associates with the corresponding traffic condition; (2) the physical distance from the current node to the end node $d_{r_i \to r_d}$; (3) the embedding of the adjacent nodes of current node e_{r_a}, which encodes the spatio-temporal traffic features explored in Section 4.2; and (4) the global traffic condition f_{t_i}, which captures the changes in road conditions in the road network. Formally, we have

$$\boldsymbol{E}_q = \mathcal{F}_e([t_i, d_{r_i \to r_d}, e_{r_i}, e_{r_d}, g_{r_i}, f_{t_i}]), \tag{8}$$

$$\hat{\mathrm{Y}} = \mathcal{F}_h(\boldsymbol{E}_q), \tag{9}$$

where $\hat{y}$ is the estimated travel time. $\mathcal{F}_e(\cdot)$, $\mathcal{F}_h(\cdot)$ are designed as two different MLPs. $\boldsymbol{E}_q$ represent the hidden representation of the query $q = (t_i, r_i, r_d)$. The operator $[\cdot, \cdot]$ represents the concatenation.

Main loss: We calculate the difference between the estimated travel time $\hat{y}$ and the actual travel time y using the **Mean Absolute Percentage (MAP)** loss and minimize it by

$$\mathcal{L}_1 = MAP(y, \hat{y}). \tag{10}$$

Auxiliary loss: During the training phase, we employ the time-dependent Dijkstra algorithm to obtain the fastest path and its travel time for each OD query. The path is a sequence of consecutive road segment IDs. We use the model described in Section 4.2 to obtain the spatio-temporal representation of each road segment and then encode the representation sequence using a Long Short-Term Memory network to derive the hidden representation of the fastest path, which is denoted as $\boldsymbol{E}_r$. Next, we compute the Euclidean distance between $\boldsymbol{E}_r$ and $\boldsymbol{E}_q$. To make the representation of the query close to the representation of the path, we minimize the above distance by the following expression:

$$\mathcal{L}_2 = \sqrt{\sum_j (\boldsymbol{E}_r[j] - \boldsymbol{E}_q[j])^2}. \tag{11}$$

Following Reference [24], we introduce dynamic adaptive weights to combine the two loss functions (i.e., auxiliary loss and main loss) and obtain the final loss for our model. This helps to balance the contribution of both losses to the overall training process and improves the accuracy of our predictions,

$$\mathcal{L} = \mathcal{L}_1 + \alpha * \mathcal{L}_2, \tag{12}$$

where α is a hyperparameter used to balance the main and auxiliary losses.

4.4 The Fastest Route Search Algorithm

In this part, we will describe how the two modules illustrated above are integrated into the A* search algorithm, which helps to effectively find the fastest routes for users' queries.

In the A* search algorithm, when we expand the child nodes of the current node, we need to calculate the observable cost and estimated cost of each expanded node. Among these costs, the observable cost of a child node consists of two parts. The first part is the searched path, that is, the travel time from the starting point to the candidate node, this part is a constant. The other part is from the current node to each extended child node, which is computed by the congestion-wise spatio-temporal prediction model described in Section 4.2. This model can capture the dynamic traffic information in the road network to accurately predict travel times for each road segment. After estimating the cost of each extended node, we input the extended node ID, destination node ID, and context information into the path-aided OD travel-time estimation module, as shown in Section 4.3. This allows us to obtain the estimated travel time from the extended node to the destination. which serves as a valuable heuristic to guide the A* search algorithm.

By introducing the congestion-aware segment travel-time prediction model to compute $g(n)$ and the path-aided OD travel-time estimation model to infer $h(n)$, our model can more effectively find the path with the shortest travel time. The detailed procedure of our approach is shown in Algorithm 1. In the search algorithm, we construct two priority queues, called closed set (C) and open set (O). At each step, the node l^* with minimum cost is removed from the open set and added to the closed set. We take the neighbor nodes of node l^* as the candidate nodes. For all candidate nodes, we use Equation (3) to compute the value of the observable cost $g(\cdot)$ function and use Equation (8) to compute the value of the estimable cost $h(\cdot)$ function. Finally, these two cost

values are added together to form the final evaluation cost. We estimate the observable cost ($G[l^{'}]$) of each neighbor l' of l^* from the prediction network. The overall cost ($F[l^{'}]$) is also updated by using the value network to obtain new estimated costs. We find the node with the lowest F value in the queue as the next target node until the end of the search is reached.

ALGORITHM 1: The search algorithm for our model.

Input: Starting point l_s, end point l_d, departure time t_q
Output: Fastest path p_l, time cost t_p

begin
Initialize O, C, F, G;
while O is not empty **do**
Obtain location $l^* \leftarrow O$.pop() ▷ location with the lowest $F[l_d]$
If $l^* = l_d$ **then**
$t^* \leftarrow F[l_d]$
Backtracking recursion to find a path p_l from l_s to l_d
return the derived route p_l from l_s to l_d and shortest travel time t^*
end
O.remove(l^*)
C.add(l^*)
for neighbor $l^{'} \in \mathcal{L}_{l^*}$ **do**
if $l^{'} \in C$ **then**
continue
end
$G' \leftarrow G[l_c] + \underline{g(l_s \rightarrow l')}$ ▷ Computed by Equation (3)
if $l' \notin O$ **then**
O.add(l')
else if $G' \geq G[l']$ **then**
continue
$G[l'] \leftarrow G'$
$F[l'] \leftarrow G[l'] + \underline{h(l' \rightarrow l_d)}$ ▷ Computed by Equation (8)
end
end
end

5 EXPERIMENTS

In this section, we conduct extensive experiments to verify the effectiveness of our proposed model based on two real-world traffic datasets. There are the following research questions to answer:

- How does our model perform compared with the state-of-the-art baselines in the task of fastest route search?
- How does the complexity of our model compare to that of the baselines?
- How does our model perform without different modules?
- How does the congestion-aware segment travel-time prediction module perform in the absence of certain sub-modules?
- How does our model more effectively search the fastest route compared with the widely used approach?

5.1 Experimental Setup

5.1.1 Dataset. In our experiment, we use two real-world datasets to test the performance of the proposed model. The statistics of the two datasets are shown in Table 1.

— *BJ-TT*: This dataset is collected from Baidu Map's open interfaces,[1] which contains the travel time of road segments in Beijing from April 6, 2022 to April 21, 2022. Its sampling period is 5 minutes. We use the OpenStreetMap[2] to obtain the road network, and matched the BJ-TT dataset to the road network using the location information of road segments.
— *Q-Traffic*: This dataset is released in Reference [23], which records the traffic speed of the road segments in Beijing with a sampling period of 15 minutes. The time range is from April 1, 2017 to May 31, 2017.

5.1.2 Evaluation Metrics. For this experiment, we use two metrics to measure the performance of our model:

$$FR1 = \frac{N_{corr}}{N_{query}}, \tag{13}$$

$$FR2 = \frac{|y - \hat{y}|}{\hat{y}}, \tag{14}$$

where N_{corr} is the number of the routes with the shortest travel time. N_{query} is the number of queries. y represents the travel time calculated by our model or the baselines described in the following section, and $\hat{y}$ represents the travel time calculated by the Dijkstra algorithm over the time-dependent graph built on the real data.

5.1.3 Experimental Settings. We split the datasets with 70% for training, 10% for validation, and 20% for testing in a chronological manner. Note that we use the dataset Q-Traffic similarly to References [23, 39]. Since the data setting is not disclosed in their works clearly, we are unable to fully replicate it. However, we can still make comparisons by using similar datasets that share similar characteristics or properties. For the test dataset, we classify the queries into three types according to the distance between the source node to the destination, i.e., short queries, medium queries, and long queries. In the BJ-TT dataset, we define the short queries as a distance smaller than 5 km, medium queries as a distance between 5 and 10 km, long queries as a distance larger than 10 km. In the Q-Traffic dataset, we define the short queries as the distance between 4 and 5 km, medium queries as the distance between 5 and 6 km, long queries as distance larger than 6 km. For the travel-time prediction module, the layers of GCN are set to two. For the shortest travel-time module, the layers of MLP are set to three. We implement our model using Python with Pytorch 1.9 on NVIDIA 3080 GPU.[3]

5.1.4 Baselines. In our experiment, we use the following methods as baselines:

— *STATIC*: This baseline is to find the shortest path based on a static travel-time graph built on the traffic information at the departure time of a query. We use the Dijkstra algorithm to implement it.
— *IAFP* [17]: This method uses the A* search algorithm to find the fastest path on a time-dependent graph based on historical data.

[1] https://map.baidu.com
[2] https://www.openstreetmap.org
[3] Code is available at https://github.com/tsinghua-fib-lab/TKDD2023-CSTGCN-Astar

Table 1. Dataset Statistics

Datasets	Interval	#Road segments	#Locations	Time Range
BJ-TT	5 minutes	486	512	04/06/2022–04/21/2022
Q-Traffic	15 minutes	1,161	1,052	04/01/2017–05/31/2017

— *ARIMA*: This baseline uses **AutoRegressive Integrated Moving Average (ARIMA)** model [1] to predict the travel time of road segments. Based on the prediction results, we use the Dijkstra algorithm to find the fastest path.
— *SVR*: Support Vector Regression [38] is used to predict the travel time of road segments. With the prediction results, we use the Dijkstra algorithm to find the fastest path.
— *Graph WaveNet* [40]: This method uses a graph convolutional network and temporal convolutional network to predict the travel time of road segments. Based on the prediction results, we use the Dijkstra algorithm to find the fastest path.
— *GMAN* [49]: The model utilizes a graph multi-attention network with an encoder-decoder architecture. Additionally, the Dijkstra algorithm is utilized to determine the fastest path.
— *NASF* [39]: This method adopts neural networks for improving A* search algorithm in fastest route recommendation task.
— *ASNN* [31]: This method combines adaptive graph convolutional recurrent network and multi-task representation learning to estimate travel time in fastest route recommendation task.

5.2 Overall Performance (RQ1)

We compare the proposed model with the state-of-the-art baselines, and the results are shown in Table 2. Our model consistently outperforms all the baselines, which demonstrates its effectiveness.

In particular, we observe that compared with the IAFP method using the historical traffic data, the STATIC method based on the current traffic information can achieve better performance. This is because while there are significant variations in the travel time of road segments at different time slots during the day, such variations are small in a relatively short time. Thus, for a fast route search, it is necessary to consider real-time traffic conditions.

Prediction-based baselines, such as ARIMA, SVR, Graph WaveNet, GMAN, NASF, and ASNN, achieve better performance when compared with the STATIC and IAFP methods. This further shows that accurate travel-time prediction for road segments is crucial to the fastest route search task. Among the prediction-based baselines, the ASNN model exhibits the best performance. Specifically, from Table 2, it is evident that the segment travel-time prediction performance ranks as ASNN>NASF>GMAN>Graph WaveNet>SVR>ARIMA, which is consistent with the overall performance of the fastest route search using the corresponding prediction model. This indicates that accurately predicting the travel time can improve the effectiveness of the fastest route search.

Our model fully considers the influence of traffic congestion on the travel time of road segments. We also effectively utilize affiliated path information for learning model parameters in OD travel-time estimation. Through accurate prediction of the $g(\cdot)$ and $h(\cdot)$ functions in the A* algorithm, the model's performance is further enhanced.

5.3 Model Complexity (RQ2)

To provide a better understanding of the proposed model, we conduct a comparison of its complexity with that of various baselines. We discuss model complexity from two perspectives. First, in terms of theoretical analysis, we calculate the number of model parameters. As shown in Table 3, our model has fewer parameters than other baselines except Graph Wavenet. However, despite

Table 2. Overall Performance of Our Model and Several Baselines over Two Datasets

Dataset	Metric	BJ-TT			Q-Traffic		
	Length	short	medium	long	short	medium	long
FR1	STATIC	0.913	0.813	0.773	0.833	0.743	0.683
	IAFP	0.906	0.806	0.756	0.82	0.73	0.66
	ARIMA	0.92	0.84	0.78	0.85	0.75	0.69
	SVR	0.93	0.846	0.796	0.86	0.75	0.69
	Graph WaveNet	0.936	0.886	0.826	0.863	0.766	0.72
	GMAN	0.943	0.89	0.836	0.86	0.773	0.7
	NASF	0.943	0.9	0.833	0.87	0.78	0.73
	ASNN	0.95	0.906	0.85	0.883	0.78	0.753
	Ours	**0.96**	**0.916**	**0.873**	**0.91**	**0.81**	**0.766**
FR2	STATIC	0.00404	0.01057	0.01299	0.00091	0.00169	0.00181
	IAFP	0.00473	0.01238	0.01346	0.00103	0.00178	0.00197
	ARIMA	0.00324	0.00438	0.00504	0.00077	0.00144	0.00177
	SVR	0.00182	0.00333	0.00469	0.00072	0.00138	0.00168
	Graph WaveNet	0.0021	0.0027	0.00317	0.00060	0.00107	0.00156
	GMAN	0.00203	0.00213	0.00286	0.00064	0.00127	0.00156
	NASF	0.00164	0.00207	0.00297	0.00068	0.00135	0.00139
	ASNN	0.0012	0.00156	0.00223	0.00047	0.00126	**0.00131**
	Ours	**0.00102**	**0.00137**	**0.00183**	**0.00033**	**0.00101**	0.00145

Bold highlights the best results, while underlining indicates the second best performance.

Table 3. The Model Complexity Comparison with Baselines

Dataset	# Parameters	Training Time (s)	Inference Time (s)
Graph Wavenet	0.27M	7.82	0.44
GMAN	5.37M	244.06	22.4
NASF	2.84M	92.21	8.81
ASNN	1.41M	72.90	6.12
Ours	1.35M	12.6	0.48

M: million (10^6)

having fewer parameters, Graph Wavenet's prediction performance is notably worse than that of our model. This indicates that our model achieves a good tradeoff between performance and efficiency. Second, from an empirical perspective, we measure the average training and inference time per epoch. The results indicate that although our model integrates multiple deep modules, it remains efficient in both training and inference time.

5.4 Ablation Study (RQ3 and RQ4)

5.4.1 Effect of Model Modules. First, we analyze the effect of different modules on the search performance. We use different variants of our model as follows: (1) w/o SCEM, which does not model the congestion features; (2) w/o GCN, which removes the graph convolutional network; and (3) w/o TA, which removes the trajectory auxiliary loss. Based on our ablation experiments, we find that all the modules are effective in improving the overall performance of our search model. Especially, it is observed that the removal of the auxiliary loss during training significantly degrades the model performance. This indicates that it is more important to use the affiliated path information in the encoding of the query during the training phase.

Table 4. Performance Comparison of Different Modules in Our Model over Q-Traffic Dataset

	Query	Short	Medium	Long
FR1	w/o GCN	0.856	0.773	0.69
	w/o SCEM	0.87	0.763	0.723
	w/o TA	0.823	0.726	0.653
	Ours	**0.91**	**0.81**	**0.766**
FR2	w/o GCN	0.00081	0.00144	0.00197
	w/o SCEM	0.00050	0.00141	0.00127
	w/o TA	0.00099	0.00186	0.00213
	Ours	**0.00033**	**0.00101**	**0.00145**

Bold is used to highlight the best results. Based on our ablation experiments, we find that all the modules are effective in improving the overall performance of our search model.

Table 5. Prediction Performance of the Congestion-aware Road Segment Travel Time Estimation Module in Terms of MAPE

Dataset	**BJ-TT**	**Q-Traffic**
w/o GCN	0.10187	0.09651
w/o SCEM	0.10049	0.09412
w/o Daily Period	0.09963	0.09317
w/o Weekly Period	0.09593	0.09175
Ours	0.09503	0.09092

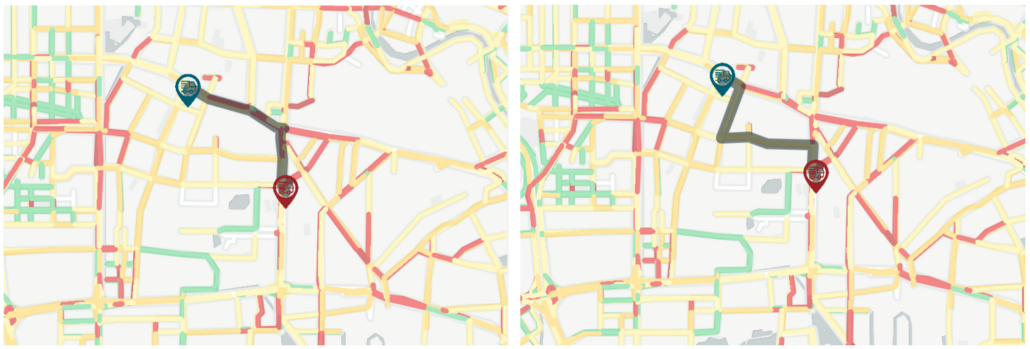
5.4.2 Effect of Road Segment Travel Time Estimation Module. For the congestion-aware segment travel-time estimation module, to verify the effect of different sub-modules on the prediction performance, we design different variants of this module: (1) *w/o SCEM*, which does not model the sequence of congestion events; (2) *w/o Daily Period*, which does not introduce the daily-periodic traffic information; (3) *w/o Weekly Period*, which removes the weekly-periodic traffic information; and (4) *w/o GCN*, which does not explore the spatial correlations by removing the graph convolution network. We use **Mean Absolute Percentage Error (MAPE)** metric to evaluate the performance of different variants. MAPE is defined as the average percentage difference between predicted and actual values. It offers a meaningful insight into the accuracy of predictions and is widely utilized in similar studies within the travel-time estimation domain [7, 8, 15, 32, 39].

Based on our empirical study, as shown in Table 5, we find that GCN plays a significant role in improving the performance of travel-time prediction. Specifically, incorporating topological information between road segments is instrumental to achieving the best predictive results. Moreover, our experimental results demonstrate that incorporating congestion events can enhance model performance for future segment travel-time prediction. Additionally, we verify the importance of capturing daily and weekly periodicity in improving accuracy. Specifically, daily periodicity yielded a higher impact on forecast performance than weekly periodicity. Thus, we conclude that the sub-modules are well designed in our congestion-aware segment travel-time estimation module.

5.4.3 Effect of Path-aided Module. In our research, we conducted experiments to evaluate the effectiveness of the **path-aided module (PAM)**. We compared its performance in terms of several metrics, including FR1, FR2, running time, the total time taken to complete all path queries, as well as the number of searched nodes per query using the A* algorithm.

Table 6. Overall Performance of the Path Aided Module over BJ-TT Datasets

Metric	**Ours**			**w/o PAM**		
Query	Short	Medium	Long	Short	Medium	Long
FR1	0.96	0.916	0.873	0.97	0.926	0.88
FR2	0.00102	0.00137	0.00183	0.00095	0.00129	0.00178
Running Time (s)	12.86	42.95	194.41	109.23	806.27	25.94
# Searched Nodes	594	2,237	8,167	2,189	10,418	37,010



(a) Dijkstra algorithm (b) Ours

Fig. 5. Two routes searched by the Dijkstra algorithm and our method.

As shown in Table 6, we find that removing the path-aided module results in a slight improvement in the quality of query results. However, it greatly increases query time and requires a larger number of nodes to be searched. For example, when dealing with long paths, the number of searched nodes and the running time increase by 353.16% and 314.73% compared to scenarios without the path-aided module, respectively. These results show that the path-aided module plays an important role in enhancing computational efficiency, without compromising the accuracy of query results.

5.5 Case Study (RQ5)

In this subsection, we give a showcase to evaluate the effectiveness of our model in the task of fastest route search. This case is to utilize the search algorithm to find the fastest path under the dynamic traffic environment. We visualize the search results of our model and the Dijkstra algorithm in Figure 5. The color of road segments represents the traffic condition level. The green color indicates that the road segment is unobstructed, and the red color indicates the congested condition. The blue marker represents the starting point of the query, and the red one is the end of the query. We show an example of finding the fastest route in Beijing at 9 AM. Figure 5(a) and Figure 5(b) are the results obtained by using static Dijkstra search and our algorithm, respectively. It can be seen that our algorithm can effectively avoid congested roads. Meanwhile, the travel time of the routes searched by the baseline and our algorithm is 734 and 729 s, respectively. This further demonstrates that our method can find the fastest route with the shortest travel time.

6 CONCLUSION

We proposed a congestion-aware spatio-temporal graph convolutional network-based A* search algorithm to implement fast route planning. We developed two modules to enhance the A* search algorithm, focusing on learning its key functions. The first is a congestion-aware neural

network that models the influence of traffic congestion on travel-time estimation. The second is a path-aided neural network that estimates travel time for an OD query by considering the relationship between the OD representation and the spatio-temporal representation of the path. Finally, we fuse the output results of both modules to set the cost function in the A* algorithm, which effectively guides the route search. Extensive experimental results showed that the proposed algorithm can achieve superior performance compared with the state-of-the-art baselines. For future work, we aim to deploy this algorithm in real-world map service applications.

REFERENCES

[1] GEP Box and G. M. Jenkins. 2015. Time series analysis: Forecasting and control. John Wiley & Sons, (2015).

[2] Cen Chen, Kenli Li, Sin G Teo, Xiaofeng Zou, Keqin Li, and Zeng Zeng. 2020. Citywide traffic flow prediction based on multiple gated spatio-temporal convolutional neural networks. *ACM Trans. Knowl. Discov. Data* 14, 4 (2020), 1–23.

[3] Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *International Conference on Machine Learning*. PMLR, 933–941.

[4] Austin Derrow-Pinion, Jennifer She, David Wong, Oliver Lange, Todd Hester, Luis Perez, Marc Nunkesser, Seongjae Lee, Xueying Guo, Brett Wiltshire, et al. 2021. Eta prediction with graph neural networks in google maps. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 3767–3776.

[5] Bolin Ding, Jeffrey Xu Yu, and Lu Qin. 2008. Finding time-dependent shortest paths over large graphs. In *Proceedings of the 11th International Conference on Extending Database Technology: Advances in Database Technology*. 205–216.

[6] Xiaomin Fang, Jizhou Huang, Fan Wang, Lihang Liu, Yibo Sun, and Haifeng Wang. 2021. Ssml: Self-supervised meta-learner for en route travel time estimation at baidu maps. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 2840–2848.

[7] Xiaomin Fang, Jizhou Huang, Fan Wang, Lingke Zeng, Haijin Liang, and Haifeng Wang. 2020. Constgat: Contextual spatial-temporal graph attention network for travel time estimation at baidu maps. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2697–2705.

[8] Kun Fu, Fanlin Meng, Jieping Ye, and Zheng Wang. 2020. CompactETA: A fast inference system for travel time prediction. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 3337–3345.

[9] Hector Gonzalez, Jiawei Han, Xiaolei Li, Margaret Myslinska, and John Paul Sondag. 2007. Adaptive fastest path computation on a road network: A traffic mining approach. In *Proceedings of the 33rd International Conference on Very Large Data Bases*. 794–805.

[10] Liangzhe Han, Bowen Du, Leilei Sun, Yanjie Fu, Yisheng Lv, and Hui Xiong. 2021. Dynamic and multi-faceted spatio-temporal deep learning for traffic speed forecasting. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 547–555.

[11] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Sci. Cybernet.* 4, 2 (1968), 100–107.

[12] Yu Huang, Josh Jia-Ching Ying, Philip S. Yu, and Vincent S. Tseng. 2020. Dynamic graph mining for multi-weight multi-destination route planning with deadlines constraints. *ACM Trans. Knowl. Discov. Data* 15, 1 (Dec. 2020), 3:1–3:32. https://doi.org/10.1145/3412363

[13] E. Jenelius and H. N. Koutsopoulos. 2013. Travel time estimation for urban road networks using low frequency probe vehicle data. *Transportation Research Part B: Methodological* 53 (2013), 64–81.

[14] Guangyin Jin, Huan Yan, Fuxian Li, Yong Li, and Jincai Huang. 2021. Hierarchical neural architecture search for travel time estimation. In *Proceedings of the 29th International Conference on Advances in Geographic Information Systems*. 91–94.

[15] G. Jin, H. Yan, F. Li, et al. 2023. Dual graph convolution architecture search for travel time estimation. *ACM Transactions on Intelligent Systems and Technology* 14, 4 (2023), 1–23.

[16] Ishan Jindal, Xuewen Chen, Matthew Nokleby, Jieping Ye, et al. 2017. A unified neural network approach for estimating travel time and distance for a taxi trip. arXiv:1710.04350. Retrieved from https://arxiv.org/abs/1710.04350

[17] Evangelos Kanoulas, Yang Du, Tian Xia, and Donghui Zhang. 2006. Finding fastest paths on a road network with speed patterns. In *Proceedings of the 22nd International Conference on Data Engineering (ICDE'06)*. IEEE, 10–10.

[18] F. Li, J. Feng, H. Yan, et al. 2023. Dynamic graph convolutional recurrent network for traffic prediction: Benchmark and solution. *ACM Transactions on Knowledge Discovery from Data* 17, 1 (2023), 1–21.

[19] Fuxian Li, Huan Yan, Hongjie Sui, Deng Wang, Fan Zuo, Yue Liu, Yong Li, and Depeng Jin. 2023. Periodic shift and event-aware spatio-temporal graph convolutional network for traffic congestion prediction. In *Proceedings of the 31st SIGSPATIAL International Conference on Advances in Geographic Information Systems*. 1–10.

[20] H. Li, D. Jin, X. Li, et al. 2023. Dmgf-net: an efficient dynamic multi-graph fusion network for traffic prediction. *ACM Transactions on Knowledge Discovery from Data* 17, 7 (2023), 1–19.
[21] L. Li, S. Wang, and X. Zhou. 2020. Fastest path query answering using time-dependent hop-labeling in road network. *IEEE Transactions on Knowledge and Data Engineering* 34, 1 (2020), 300–313.
[22] Yaguang Li, Kun Fu, Zheng Wang, Cyrus Shahabi, Jieping Ye, and Yan Liu. 2018. Multi-task representation learning for travel time estimation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1695–1704.
[23] Binbing Liao, Jingqing Zhang, Chao Wu, Douglas McIlwraith, Tong Chen, Shengwen Yang, Yike Guo, and Fei Wu. 2018. Deep sequence learning with auxiliary information for traffic prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 537–546.
[24] Lukas Liebel and Marco Körner. 2018. Auxiliary tasks in multi-task learning. arXiv:1805.06334. Retrieved from https://arxiv.org/abs/1805.06334
[25] Giacomo Nannicini, Daniel Delling, Leo Liberti, and Dominik Schultes. 2008. Bidirectional A search for time-dependent fast paths. In *International Workshop on Experimental and Efficient Algorithms*. Springer, 334–346.
[26] Xiaoguang Niu, Ying Zhu, Qingqing Cao, Xining Zhang, Wei Xie, and Kun Zheng. 2015. An online-traffic-prediction based route finding mechanism for smart city. *Int. J. Distrib. Sens. Netw.* 11, 8 (2015), 970256.
[27] Stefano Pallottino and Maria Grazia Scutella. 1998. Shortest path algorithms in transportation models: Classical and innovative aspects. In *Equilibrium and Advanced Transportation Modelling*. Springer, 245–281.
[28] J. Rice and E. Van Zwet. 2004. A simple and effective method for predicting travel times on freeways. *IEEE Transactions on Intelligent Transportation Systems* 5, 3 (2004), 200–207.
[29] Raffi Sevlian and Ram Rajagopal. 2010. Travel time estimation using floating car data. arXiv:1012.4249. Retrieved from https://arxiv.org/abs/1012.4249
[30] Shuo Shang, Ruogu Ding, Kai Zheng, Christian S. Jensen, Panos Kalnis, and Xiaofang Zhou. 2014. Personalized trajectory matching in spatial networks. *VLDB J.* 23 (2014), 449–468.
[31] Chaoxiong Wang, Chao Li, Hai Huang, Jing Qiu, Jianfeng Qu, and Lihua Yin. 2021. ASNN-FRR: A traffic-aware neural network for fastest route recommendation. *GeoInformatica* (2021), 1–22.
[32] Dong Wang, Junbo Zhang, Wei Cao, Jian Li, and Yu Zheng. 2018. When will you arrive? Estimating travel time based on deep neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.
[33] H. Wang, X. Tang, Y. H. Kuo, et al. 2019. A simple baseline for travel time estimation using large-scale trip data. *ACM Transactions on Intelligent Systems and Technology (TIST)* 10, 2 (2019), 1–22.
[34] Mudan Wang, Huan Yan, Huandong Wang, Yong Li, and Depeng Jin. 2023. Contagion process guided cross-scale spatio-temporal graph neural network for traffic congestion prediction. In *Proceedings of the 31st ACM International Conference on Advances in Geographic Information Systems*. 1–11.
[35] Y. Wang and M. Papageorgiou. 2005. Real-time freeway traffic state estimation based on extended Kalman filter: A general approach. *Transportation Research Part B: Methodological* 39, 2 (2005), 141–167.
[36] Yilun Wang, Yu Zheng, and Yexiang Xue. 2014. Travel time estimation of a path using sparse trajectories. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 25–34.
[37] Ling-Yin Wei, Yu Zheng, and Wen-Chih Peng. 2012. Constructing popular routes from uncertain trajectories. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 195–203.
[38] C. H. Wu, J. M. Ho, and D. T. Lee. 2004. Travel-time prediction with support vector regression. *IEEE Transactions on Intelligent Transportation Systems* 5, 4 (2004), 276–281.
[39] Ning Wu, Jingyuan Wang, Wayne Xin Zhao, and Yang Jin. 2019. Learning to effectively estimate the travel time for fastest route recommendation. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 1923–1932.
[40] Z. Wu, S. Pan, G. Long, et al. 2019. Graph wavenet for deep spatial-temporal graph modeling. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence* (2019), 1907–1913.
[41] S. Xu, R. Zhang, W. Cheng, et al. 2022. Mtlm: A multi-task learning model for travel time estimation. *GeoInformatica*, 26, 2 (2022), 379–395.
[42] Y. Xu, L. Han, T. Zhu, et al. 2023. Generic dynamic graph convolutional network for traffic flow forecasting. *Information Fusion* 100 (2023), 1–12.
[43] Huan Yan, Guangyin Jin, Deng Wang, Yue Liu, and Yong Li. 2022. Jointly modeling intersections and road segments for travel time estimation via dual graph convolutional networks. In *International Conference on Spatial Data and Intelligence*. Springer, 19–34.
[44] Huan Yan, Guangyin Jin, Deng Wang, Yue Liu, and Yong Li. 2023. Jointly modeling intersections and road segments for travel time estimation via dual graph convolutional networks. In *Spatial Data and Intelligence: Third International Conference, SpatialDI 2022, Revised Selected Papers*. Springer, 19–34.

[45] H. Yuan, G. Li, and Z. Bao, et al. 2020. Effective travel time estimation: When historical trajectories over road networks matter. In *Proceedings of the 2020 ACM Sigmod International Conference on Management of Data* (2020), 2135–2149.
[46] Jing Yuan, Yu Zheng, Xing Xie, and Guangzhong Sun. 2011. Driving with knowledge from the physical world. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 316–324.
[47] J. Yuan, Y. Zheng, X. Xie, et al. 2011. T-drive: Enhancing driving directions with taxi drivers' intelligence. *IEEE Transactions on Knowledge and Data Engineering* 25, 1 (2011), 220–232.
[48] Jing Yuan, Yu Zheng, Chengyang Zhang, Wenlei Xie, Xing Xie, Guangzhong Sun, and Yan Huang. 2010. T-drive: Driving directions based on taxi trajectories. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*. 99–108.
[49] Chuanpan Zheng, Xiaoliang Fan, Cheng Wang, and Jianzhong Qi. 2020. Gman: A graph multi-attention network for traffic prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 1234–1241.

Received 12 March 2023; revised 30 December 2023; accepted 31 March 2024