Abstract Traveling time forecasting, the core component in GPS navigation systems and taxi-hailing apps, has attracted widespread attention. The conventional approaches mainly focus on the routing algorithms on a given stationary road network, ignoring the real-time dynamic change of traffic conditions and the relationship between adjacent road segments, which will significantly affect the planning of the optimal route. Motivated by the above observation, we propose a novel dynamic routing planning framework DTT-STG integrated with map-matching, road speed forecasting, and route planning with full consideration of the dynamic spatial and temporal dependency from a large receptive field. DTT-STG innovatively designs an angle-based map-matching algorithm to describe the direction of vehicles and utilizes a spatial diffusion and temporal attention block to capture the features from the nearby road segments. Afterward, the progressive method is exploited to calculate the traveling time and plan the shortest route dynamically in continuously changing traffic conditions. The comprehensive experiments indicate that DTT-STG achieves higher accuracy and efficiency than other off-the-shelf models.

Keywords Traveling Time Forecasting, Dynamic Route Planning, Intelligent Transportation System, Spatial Diffusion, Temporal Attention.

1 Introduction

In intelligent transportation scenarios, the integration of efficient route planning algorithms into GPS navigation systems has become a heated issue, which has received increasing attention from the research community. Many scholars have given corresponding solutions by combining probabilistic statistical [1, 2], and deep learning [3-6]. However, most route planning algorithms take the shortest travel distance as the optimization goal [3-5][7-10] rather than the shortest traveling time [11].

Besides, these models are designed on the assumption that the traffic condition remains the same as the departing time, which may not meet the urgent need of users to plan the best routes under dynamic and complex traffic conditions. To deal with this issue, the route planning models combined with traffic speed forecasting [12-19] become the mainstream paradigm. Nonetheless, the changeable traffic conditions from a dynamic perspective still lack satisfactory progress.

To facilitate this line of research, we proposed a **D**ynamic

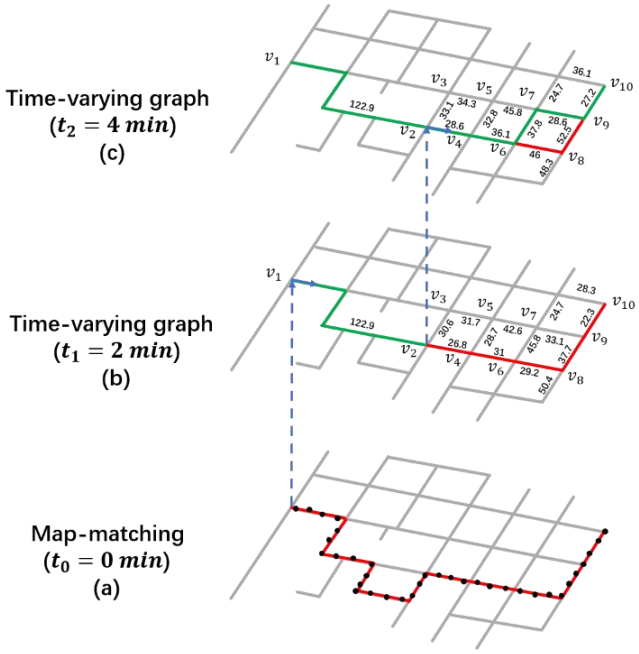


Fig. 1 An example to illustrate the process of DTT-STG model. The process of map-matching is depicted in Fig. 1(a), and the route planning is depicted in Fig. 1(b) and Fig. 1(c).

Traveling Time forecasting framework based on the Spatial and Temporal Graph convolution (DTT-STG). DTT-STG explores a self-adaptive adjacency matrix combined with diffusion convolution and attention mechanisms to capture the dynamically changing spatial-temporal attributes of the road network. In this way, the traffic flow can be forecast more accurately and efficiently. Furthermore, we explore the progressive method to calculate the traveling time and plan the shortest route dynamically in the continuously changing traffic condition. To the best of our knowledge, we are the first to apply advanced traffic flow forecasting models to dynamic route planning.

DTT-STG provides a complete solution for route planning with the shortest traveling time from a global perspective rather than merely focusing on independent sub-problems. Map-matching, road speed forecasting, and route planning are integrated into a unified framework in our proposal. DTT-STG captures the spatial and temporal dependencies of the nearby road segments, especially during the morning and night peaks while the traffic conditions are changing dramatically. Furthermore, DTT-STG is exploited to deploy as a progressive form for satisfying the changeable real-time traffic condition, since route planning is well recognized as a time and memory-consuming NP-hard problem.

DTT-STG is composed of an Angle-based Map-Matching (AMM), road speed forecasting with Diffusion Spatial and Temporal Graph convolution network (DiffSTG), and route planning with Improved Dynamic Dijkstra (IDD) modules. To better elaborate the process of DTT-STG, Fig. 1 displays a specific case of planning the shortest time path between the starting point v_1 and the ending point v_{10} with the start timestamp t_0.

First, the AMM module is implemented to align a sequence of GPS-sampled positions of the trajectory to the digital map. Thus, it is available to find the road through which these trajectories traveled, i.e., the transformation from trajectory sequence (black points in Fig.1 (a)) to navigation path sequence (red line in Fig.1 (a)). Afterward, DiffSTG module adopts a spatial and temporal convolution network to predict the road speed of each road segment every 2 minutes (the time interval can be tuned), and the traveling time of each road segment can be estimated with the predicted road speed, which is annotated on edge in Fig.1. Based on the aforementioned process, DTT-STG framework will calculate the best route with IDD progress (green line). Compared with the real trajectory derived from navigation systems (red line), DTT-STG will find out the shortest route based on the latest traffic condition using a greedy algorithm. For example, DTT-STG chooses (v_6, v_7) instead of (v_6, v_8) since the traveling time of (v_6, v_7, v_9) equals to 66.4s which is shorter than 98.5s of (v_6, v_8, v_9). Notice that, facing the changing of traffic conditions, the best path changes from (v_6, v_8, v_9) to (v_6, v_7, v_9) as the timestamp moves from t_1 to t_2, which saves about 32s (nearly 10% of the total time which is about 313.4s from v_1 to v_{10}). Therefore, DTT-STG owns the capability of planning the best route when facing real-time changeable traffic conditions. For optimization, we additionally introduce the traveling direction of vehicles for map-matching, utilize a spatial and temporal attention block to capture the features from the nearby road segments, and adopt a progressive method to calculate the traveling time for route planning dynamically in DTT-STG. Our major contributions can be summarized as follows:

- Deliver a novel Dynamic route planning framework DTT-STG with comprehensive consideration of the dynamic spatial and temporal dependencies from a large receptive field, which compresses about 10% traveling time compared with ground truth.
- Investigate the influence of vehicle moving direction information in route planning, and propose an angle-based map-matching algorithm AMM.
- Construct the temporal attention and spatial diffusion block to capture the features from the nearby road segments for traffic speed forecast.
- Explore the progressive method to calculate the traveling time and plan the shortest route dynamically in the continuously changing traffic condition.

The rest of this paper is organized as follows. Section 2 provides a review and summary of relevant literature. Section 3 formally defines the problem and relevant variants. Section 4 gives out the overview of DTT-STG framework, and subsequently elaborates on the components of DTTSTG, including map-matching, road speed estimation, and dynamic route planning. Section 5 conducts comprehensive experiments to establish discernible superiority of our proposal, and Section 6 concludes our work.

2 Related Works

Dynamic route planning based on traveling time forecast can be divided into three sub-problems: map-matching, road speed forecasting, and route planning. The related works will be elaborated on these three aspects.

2.1 Map-matching

Map-matching intends to align a sequence of observed positions on a trajectory to a digital map, which is a necessary pre-processing step in the tasks of traffic forecasting and driving behavior analysis. According to the technical characteristics, map-matching can be divided into two mainstreams, i.e., real-time map-matching and post-processing pattern matching. Real-time map-matching requires instantaneously updating users' GPS points on the road network. In contrast, post-processing map-matching focuses on mining a large number of users' historical trajectories [20].

Brakatsoulas et al. [21] designed a global matching algorithm based on metrics such as curve similarity and angle computation, which minimize the frechet distance between the trajectory and the matching road. While the method is sensitive to outliers and collection frequencies, and it is difficult to get the correct road sequence with longer time intervals.

Newson et al. [22] designed a map-matching algorithm based on Hidden Markov Model to process data with a moderate sampling rate. Lou et al. [23] designed a spatio-temporal candidate graph to find a global matching path with the highest

score, which significantly improved the matching accuracy on sparse sampling data. Chen et al. [24] proposed a dynamic planning technique to minimize the set of candidate paths for each GPS point and applied the algorithm to a large-scale trajectory data set with low sampling frequency. While current methods neglect the driving direction of vehicles, which will significantly restrict the performance in the scenarios of U-turns. Thus, we leverage the map-matching methods by introducing angle information.

2.2 Road speed forecasting

Road speed forecasting, as the backbone of Intelligent Transportation System (ITS), is of great importance for traffic regulation and risk assessment.

In the past decade, artificial intelligence models were showing an explosive growth trend in various application scenarios. Due to the complexity and unpredictability of traffic conditions like rush hours, roadway constraints, accidents, etc., urban road speed forecasting faces great challenges. Machine learning methods utilized expert knowledge to select these features for predicting future road speed. However, these models have serious drawbacks, such as ignoring the characteristic of diffusibility and spatial adjacency. Fortunately, deep learning models are able to automatically mine the features of the data [12]. Convolutional Neural Networks(CNNs) and Recurrent Neural Networks(RNNs) are two well-recognized paradigms in deep learning. Local connected structure, weight sharing, and pooling methods make CNN-based models fully capable of handling high-dimensional data. RNNs-based models have the ability to handle time-series data. Thus, some novel approaches combine the advantages of both. Liu et al. [13] designed the Conv-LSTM model, a recurrent neural network model, which is more powerful than those models that only consider CNN or RNN structure.

Recently, Graph Neural Networks (GNNs) skillfully transferred deep learning to graph data and achieved remarkable success in many scenarios [14]. Taking advantage of both external features (such as time-of-day, day-of-week, weather conditions, and POI) and topology information, GNN is also expected to inject new vitality into dynamic route planning in intelligent transportation systems. For instance, Li et al. [15] designed a diffusion convolutional recurrent neural network to capture spatial correlation by bi-directional random walking in the graph, and took the planned sampling with encoder-decoder architecture to capture temporal dependencies. Guo et al. [16] constructed a graph convolutional neural network with spectral clustering to predict road speed, which can extract node-level and region-level points information. Zhao et al. [17] integrated the graph convolutional model with the gated recurrent unit component, which can better capture the implied spatio-temporal feature information. Cai et al. [18] proposed the transformation assumption in the embedding representation phase.

However, the aforementioned methods just utilize a stationary value to set spatial relationships with the absence of consideration of the underlying road network. Thus, a potent model appointed DiffSTG is designed to devote the veritable spatial correlation according to the self-adaptive adjacency matrix. Moreover, DiffSTG combined diffusion convolution and attention mechanisms to exploit the dynamically changing spatial-temporal attributes of the road network.

2.3 Route planning

On the basis of the aforementioned map-matching and road speed forecasting procedures, route planning algorithms can be developed for calculating an optimal travel route by analyzing the traffic condition of the road network between the starting point and the destination. There are two mainstreams of studies, the traditional shortest path planning algorithms, and the intelligent planning algorithms. Traditional algorithms include the classic Dijkstra [25, 26], A* [27, 28] , Floyd Algorithm [29]. Since then, many optimized versions have emerged. Zhu et al. [7] used the back-labeled Dijkstra algorithm combined with breadth-first search, stack, and queue data structures to verify the time efficiency of the optimal path search structures. Probabilistic statistical models [1, 2] are another branch of traditional models. Pedersen et al. [1] and Guo et al. [2] explore the probabilistic budget model to get the optimal route within a given time budget, and the optimal route has higher probability with the least travel cost. We also adopt the probabilistic budget model and further consider the impact of dynamic traffic states and probe the spatial-temporal convolutional neural networks to capture the spatio-temporal dependence characteristics, thus the traffic flow can be forecast more accurately.

Intelligent algorithms include deep neural network models and reinforcement learning models. Liu et al. [6] combined reinforcement learning with A* algorithm and proposed a route planning algorithm capable of handling states such as traffic control. Zheng et al. [3] designed a novel model for road selection based on graph convolutional networks and the predicted paths are similar to the paths identified by experts. Liu et al. [5] proposed a cloud-based mobility framework to estimate

travel costs and find out popular routes. Recently, models considering more complicated application scenarios have been proposed. Li et al. [8] considered the diverse travel needs of different users, combined the Polychromatic Sets theory, and performed user-centered route planning. Ning et al. [9] considered the route planning problem in terms of urban road network structure, bus stop design, user capacity, etc, which achieved good performance using real data sets for rush hours and non-rush hours. Yang et al. [10] designed a Situational Road Network (SRN) model to search for the shortest traveling path in emergency situations. However, it is hard to deal with the dynamic planning situation within a time window Δt, since Δt is a stationary value, which does not have practical planning significance.

To summarize the previous research, there are none of the above studies consider the changeable traffic conditions from a dynamic perspective. Therefore, we explore the progressive method to calculate the traveling time and plan the shortest route dynamically in the continuously changing traffic condition.

3 Problem statement and definitions

In this section, we first formalize the research problem. Then frequently used notations and properties are defined.

3.1 Problem Statement

Dynamic Route Planning (DRP) is represented as a 3-tuples $DRP(p_{start}, p_{end}, t_{start})$, where $p_{start}, p_{end}, t_{start}$ stands for source, destination and the departure time of a route planning query, respectively. The DRP query aims to find out an optimal route $route_{opt} = (p_{start} \rightarrow v_1 \rightarrow ... \rightarrow v_k \rightarrow p_{end})$ to minimize the traveling time under a given confidence threshold τ, i.e., $route_{opt}.cost \leq route_i.cost$, for $\forall route_i \neq route_{opt}$, both $route_i$ and $route_{opt}$ satisfy the confidence probability threshold τ. Formally,

$$DRP(p_{start}, p_{end}, t_{start}) = argmin_{route \in \Gamma} route.cost \quad s.t. route.conf \geq \tau \qquad (1)$$

3.2 Notation Definition

To better illustrate our proposal, Table 1 presents the frequently appeared notations, and the relevant concepts are methematically defined as follows.

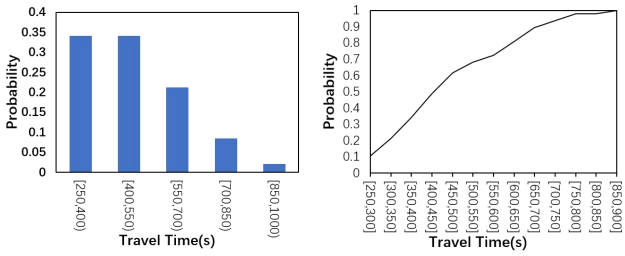
A trajectory tf_i is a historical route which is composed of a sequence of GPS location points $< p_1, p_2, ..., p_n >$, whereas $route_i$ denotes a forecasting route composed of road network vertices $< v_1, v_2, ..., v_k >$. And $route_i.cost$, $route_i.conf$ are two attributes associated with $route_i$ which describe the travel cost and confidence of $route_i$ respectively.

Definition 1 (**Road speed on a given road segment**): The road network can be regarded as a directed and edge-weighted time-dependence graph $G(V, E, T)$, where the road network vertices (i.e. a crossroad) is defined as v_i and directed edge $e_{i,j}$ represents the association of two nearby road segments v_i and v_j. Besides, $S_{e_{i,j}}^{\Delta t}$ denotes the road speed on this edge segment at time interval Δt.

Definition 2 (Potential route and traveling time): Given a routing query (including a source point, a destination point, and a departure time), a set of potential routes Γ is available for traveling, and the corresponding traveling time is defined as $route_i.cost$.

Notation	Explanation
$G(V, E, T)$	time-varying road network with node set V and edge set E at a time of Δt
v_i	vertex of road network, where $v_i \in V$
$e_{i,j}$	directed edge of road network from v_i to v_j, where $e_{i,j} \in E$
ϕ	dataset of history trajectories
tf_i	the ith trajectory, where $tf_i \in \phi$ and $tf_i =< p_1, p_2, ...p_n >$
p_i	a GPS sampled location, it can be described as a (location,timestamp) pair $p_i.location$ is the latitude and longitude, and $p_i.timestamp$ is the timestamp
$S_{e_{i,j}}^{\Delta t}$	traveling speed on $e_{i,j}$ in a time interval Δt
$route_i$	the ith route derived by a route planning algorithm, where $route_i = (p_{start} \rightarrow v_1 \rightarrow ... \rightarrow v_k \rightarrow p_{end})$
$route_i.cost$	travel time of $route_i$
$route_i.conf$	the probability of $route_i$

Table 1 Summary of the notations.



(a) Confidence histogram (b) Confidence CDF

Fig. 2 Probability (confidence) on different travel time.

Definition 3 (**Confidence of the planned Route**) : The confidence of a given route is defined as $route_i.conf$, which derives the distribution of the possible traveling cost based on the *Probabilistic Budget* model introduced in [1] and [2].

For example, as shown in Fig. 2(a), we use histograms to represent the probability of a given travel time, i.e., $route.cost$ based on the historical trajectory dataset (the dataset is

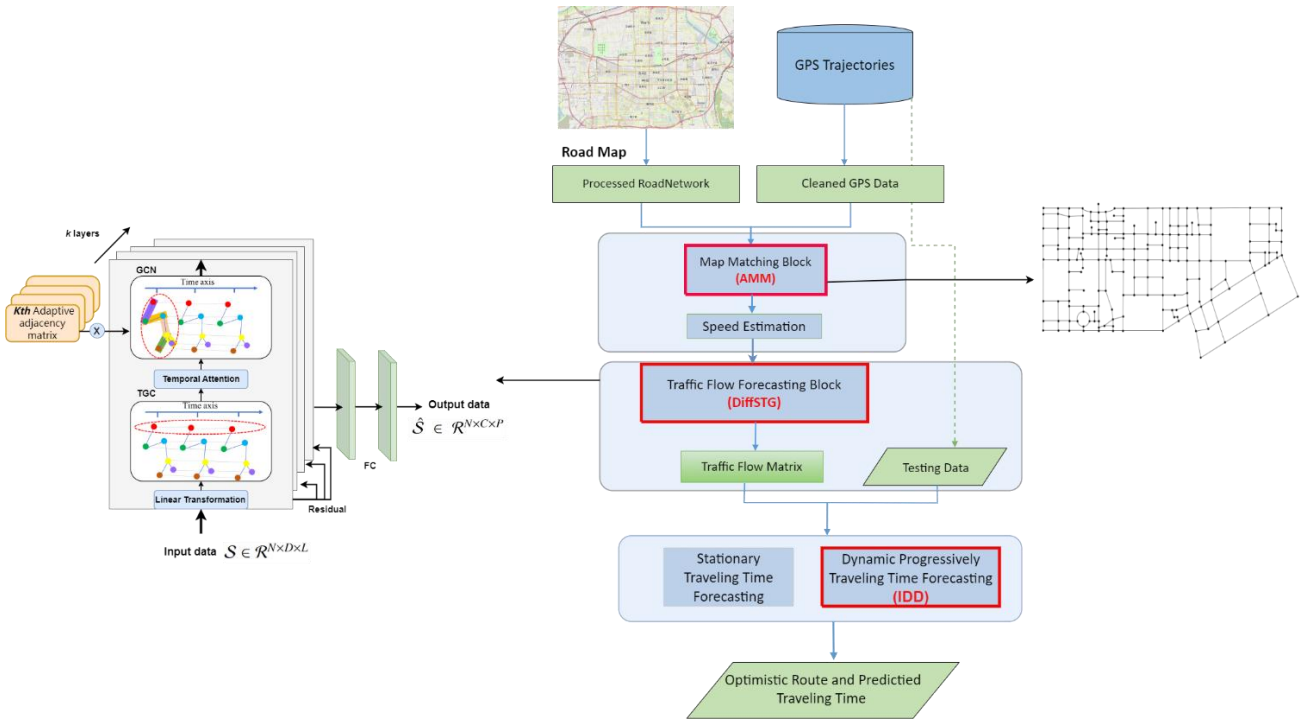


Fig. 3 Framework of DTT-STG.

introduced in Section 5.1). A histogram consists of a number of (Bucket, Probability) pairs, such that the sum over the probabilities of all pairs is 1. Fig. 2(b) shows *Cumulative Distribution Function* (CDF) based probability of a given route cost. Suppose that $route_i.cost = 450s$, it's in the second bucket of Fig. 2(a), we can get $route_i.conf = 0.34$, and from Fig. 2(b) we can derive the same confidence value.

If the output of route planning corresponds $t_1 + e_{i,j}.cost^{t_1} \leq t_2 + e_{i,j}.cost^{t_2}, t_1 < t_2$, where $e_{i,j}.cost^{t_1}$ and $e_{i,j}.cost^{t_2}$ refer to the travel time through $e_{i,j}$ at the moment t_1 or t_2. Such that the sequence of route is an optimal path.

4 Methodology

4.1 Framework of DTT-STG

From the definition of DRP, it's obvious that the query is an NP-hard problem that is impossible to be solved in polynomial time. Thus, we propose a heuristic framework DTT-STG to deal with this issue, which achieves the approximate solution within an acceptable time.

For better illustration, we briefly introduce the overview of the DTT-STG framework in Fig. 3 and Algorithm 1. First, multi-source road network data are collected from OpenStreetMap and GPS trajectories. Then, we designed a novel map-matching algorithm to project the raw trajectory data into the road network. Afterward, the historical road speed of each road segment is estimated and the missing data are imputed. And then, the deep neural network with spatial and temporal convolution layer is exploited to predict the road speed and construct the speed feature matrix for the whole road network. At last, DTT-STG dynamically forecasts the optimal route with the shortest traveling time. Notice that, the speed feature matrix is dynamically updated to capture the dynamic and complex traffic conditions when we use the dynamic progressively traveling time forecasting methods.

Algorithm 1: DTT-STG Framework

Input: $\phi, G(V, E, T), \Delta t, DTT\_query$
Output: *Route, Travel_time*

Initialize road matrix M;
for tf_i in ϕ **do**
  for j in tf_i **do**
    Execute AMM algorithm for each trajectory;
    Calculate the speed of vehicles passing each road;
for i in days **do**
  for j in hours **do**
    for k in Δt **do**
      for l in road **do**
        Update $M[i][j][k][ceil(l/\Delta t)]$
Checking the data integrity of M;
$Speed = DiffSTG(\mathcal{T}(M), data)$;
$Route, Travel\_time = IDD\ algorithm(Speed, DTT\_query)$
return $Route, Travel\_time$

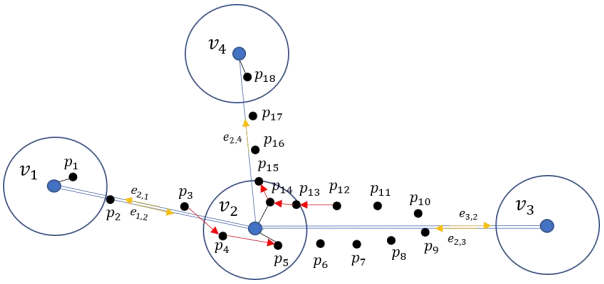


Fig. 4 An example of map-matching processing.

4.2 Map-matching

Map-matching is an essential procedure to project GPS points to the road network. The traditional distance-based map-matching algorithm first calculates the distance between trajectory points p_i and road vertex v_j in a trajectory, then sets a distance threshold θ to add vertices that satisfy $dist(p_i, v_j) \leq \theta$ to the result set. Generally speaking, a trajectory point corresponds to at most one road vertex, while a road vertex may associate with more than one trajectory point. Finally, the trajectory point sequence is converted to a road network vertex sequence. But this approach has a limitation, if we don't consider the angle information, all the nearest trajectory points of a given vertex that satisfy the distance threshold may be projected to the road in the same direction. This is usually the case when the vehicle performs a U-turn operation in the middle of the road. For example, in the case of Fig. 4, the floating vehicle enters $e_{2,3}$ at vertex v_2 and performs a U-turn operation in the middle section of $e_{2,3}$ before reaching the detection range of another road vertex. Therefore, the vertex sequence returns (v_1, v_2, v_4), and the corresponding road sequence turns out to be $(e_{1,2}, e_{2,4})$ because the node v_3 has not been added to the vertex sequence. While the true vertex sequence is $(v_1, v_2, v_3, v_2, v_4)$, which is converted to road segment sequence $(e_{1,2}, e_{2,3}, e_{3,2}, e_{2,4})$.

Motivated by the above observation, we propose an **A**ngle-based **M**ap-**M**atching algorithm (**AMM**), which considers the vehicle's traveling direction property, storing distance and direction information of trajectory points within a threshold at each road vertex. Continuing to consider the example in Fig. 4, $(p_4, p_5, p_{13}, p_{14}, p_{15})$ are the nearest point to vertex v_2 within the distance threshold, the direction angles corresponding to these points are (351, 359, 178, 162, 149). Therefore, AMM can classify these points into two categories based on the angle values, which are (p_4, p_5) belong to c_1, where the angle value is larger than 180, and the rest belong to c_2. And in these two categories, p_5 and p_{14} are the nearest point to v_2 respectively, and then calculate the directional angles of p_5 and p_{14} and the directional angles of road vertex v_2, match the directional angles of these two points with the directional angles between the adjacent vertices of $v_2 = \{v_3, v_4\}$, and find out that $\angle p_5 v_2 v_3$ can match the vehicle's running direction better than $\angle p_5 v_2 v_4$, thus $\angle p_5 v_2 v_3$ is added to the result vertex sequence. Finally AMM returns two roads $e_{2,3}$ and $e_{3,2}$. Under this circumstance, AMM can impute the missing road segment to the final road sequence, which turns out to be $(e_{1,2}, e_{2,3}, e_{3,2}, e_{2,4})$ and can match the real trajectory perfectly.

Algorithm 2: AMM Algorithm

Input: $\phi, G(V, E)$
Output: Rid, Vid

```
1  Initialize Rn, Rid, Vid;
2  for tf_i in vertices do
3    for j in φ do
4      a = distance(vertices[i], φ[j]); b = direction(φ[j]);
5      if a < threshold then
6        Rn.append([i, j, a, b])
7  Rn = Grouped by the first of Rn
8  for i in Rn do
9    Rn[i] = Grouped by the fourth of Rn
10   if len(Rn[i]) == 1 then
11     Add (Rn[i][idx][1], Rn[i][idx][0]) to (Vid, Rid);
12     Combining direction and vertices to add Rid;
13   else
14     for j in Rid do
15       Add (Rn[i][idx][1], Rn[i][idx][0]) to (Vid, Rid)
16       Combining direction and vertices to add Rid;
17 for i in Rid − 1 do
18   if Rid[i + 1] doesn't correspond adjacent relation then
19     Remove this trajectory and Break;
20 return Vid, Rid
```

The pseudo-code of AMM is formalized as Algorithm 2. Line 1 initializes the variables, where Rn stores some information about the distance less than θ between trajectory points p and vertices v, including the sequence information i in road vertices, the sequence information j in trajectory point, the distance between trajectory point and vertices a, and direction angle b of trajectory points. Rid and Vid represent the road information and vertex information of the trajectory, respectively. Lines 2 to 6 traverse the trajectory points and road network vertices, and if the distance between the trajectory points and vertices is less than the threshold, the data information of the point pair is added to Rn. The vertex sequence of Rn is grouped in line 7, which means one network vertex corresponds to one or more trajectory points. Lines 8 to 9 traverse each network vertex in Rn, and then classify the direction

information in each vertex. Lines 10 to 16 determine the roads and vertices through which the trajectory passes. Specifically, if the group number associated with a vertex is less than or equal to 1, it means that there is only one driving direction of the trajectory passing through the road network vertex. Otherwise, there are trajectory points with different driving directions. Combining the directional angle of the vertex of the adjacent and the directional angle of the trajectory points, the trajectory points of each direction nearest to the road network vertex are added to the result in turn. Considering few trajectories may lead to a huge distance between two adjacent points in the extreme condition of missing GPS signal, the outliers will be removed in lines 17 to 19. The last line returns the matching result of a single trajectory.

4.3 Road speed forecasting

4.3.1 Road speed calculation

After the map-matching procedure, we have derived the road segments that all trajectories pass through, suppose the average velocity of all trajectories points is the road speed in a certain interval Δt, we can estimate the speed of the road according to the distance and time difference between two adjacent coordinate points.

The value of the speed on a given road segment $e_{i,j}$ of a trajectory can be calculated as follows.

$$S_{p_i} = \frac{p_{i+1}.location - p_i.location}{p_{i+1}.timestamp - p_i.timestamp} \tag{2}$$

$$S_{e_{i,j}} = \frac{1}{N}\sum_{i=1}^{N} S_{p_i} \tag{3}$$

Where each p_i is one specific trajectory point with the latitude and longitude $p_i.location$, and the timestamp $p_{i+1}.location$. And N is the number of points on $e_{i,j}$ in a certain time period Δt. Considering all the trajectories that pass one road together, the velocity value of the road in a time period can be calculated as:

$$S_{e_{i,j}}^{\Delta t} = \frac{1}{|\psi|}\sum_{k=1}^{|\psi|} S_{e_{i,j}}, \psi \subseteq \phi \tag{4}$$

Where $|\psi|$ is the number of trajectories that have passed the road $e_{i,j}$ in a certain period Δt.

4.3.2 Spatial and temporal convolution for road speed prediction

Traffic topology can be viewed as a graph, where the node represents the road vertices, and the edge represents the road segment. GNN techniques transfer deep learning to non-Euclidean. Graph convolution is a pattern processing method based on the dependencies between nodes in graph networks and is the basic structure of spatio-temporal graph modeling.

In this section, we propose a novel end-to-end deep learning architecture namely DiffSTG to forecast each road speed for a given dataset of history trajectories ϕ. To illustrate the data relationship between the output of Eq.(4) and the input of DiffSTG, since DiffSTG is a convolution operation on the historical features of the vertices, we introduce a mapping $\mathcal{T}(G) \to \widehat{G}$ which could convert an edge in graph G to a point in $\widehat{G}$ according to the adjacency of the edge of G. Consequently, the output value of DiffSTG just utilize a reverse mapping $\widehat{\mathcal{T}}$ which could represent forecasting traffic speed on each edge in G.

We use $v_{i,t} \in \mathcal{R}^D$ to denote the historical speed feature of node v_i ($v_i \in \widehat{G}$) at time t, where $v_{i,t}$ is transformed from $S_{e_{i,j}}^{\Delta t}$ (the speed on road segment) by mapping function $\widehat{\mathcal{T}}$. $X_t \in \mathcal{R}^{N\times D}$ denotes the speed values of all nodes at time t, where N is the number of vertices, D is number of features (in our case, $D = 1$). $X = \{X_1, X_2, ..., X_L\}$ denotes the speed value of all the nodes over L time slices. Spatial-temporal convolution consists of the temporal gated convolution layer TGC and spatial convolution layer GCN. Moreover, to acquire the coupling of temporal dependencies, multi-head temporal attention is adopted to enhance the representation of the output features after the TGC layer. We will describe DiffSTG in detail according to the overall architecture procedure shown in left of Fig. 3.

Temporal gated convolution: Since conventional CNNs cannot handle time-series data, dilated causal convolution is exploited to capture latent patterns in the temporal dimension. Meanwhile, the prediction on each time interval takes into account the values on each interval. Considering the model may suffer from gradient explosion and complex structure when dealing with long time series data, dilated convolution structure is introduced to deal with such defects. Dilated convolution has a fixed hop value between each layer, which can exponentially increase the receptive fields of the model and the application area of the filter. Dilated causal convolution combines the advantages of both enabling a 1-dimensional structure for processing time series data while controlling the complexity of the model and improving the model's adaptability [30]. The mathematical expression for the dilated causal convolution at step t is as Eq.(5).

$$x * \boldsymbol{f}(t) = \frac{1}{M}\sum_{s=0}^{K-1} \boldsymbol{f}(s)\boldsymbol{x}(t - u \times s) \quad (5)$$

where K denotes the number of model implied layers, $f \in \mathcal{R}^K$ is a filter, $*$ denotes convolution operation, and u is the dilation factor controlling the jump distance. $x \in \mathcal{R}^L$ is a subset of input data $X \in \mathcal{R}^{N\times D\times L}$, it represents one dimension sequence that obtained from input followed by a linear transformation layer.

Besides, a gated mechanism is used to learn long-range temporal dependencies. In summary, the computation in temporal gated convolution is written as Eq.(6):

$$TGC(X) = tan\,h\,(\Phi_1 * X) \odot sigmoid(\Phi_2 * X) \quad (6)$$

where $TGC(X) \in \mathcal{R}^{N\times H\times L}$ is the output of temporal gated convolution, L is the sequence length, H is the hidden dimension of TGC, $sigmoid$ and $tanh$ are the two types of activate functions, Φ_1 and Φ_2 are model parameters.

To acquire the coupling of temporal dependencies, an attention mechanism is applied after the temporal convolution layer to enhance the derived representation of temporal features. More specifically, a mapping function computes attention vectors output through a dot inner product for a query denote as Q and a set of pairs of key-value denote as K and V. Then we get the sum of each value multiple corresponding to a weight. Multi-head attention works by implementing multiple attention modules in parallel using multiple different versions of the same query, thus allowing the attention model to introduce more feature information in the context vector computation [31]. The Eq.(7) shows the process of acquiring attention score for a node in different time Δt. in the temporal attention layer, Q, K, and V equals $TGC(X)$.

$$Attention(Q,K,V) = Softmax\left(\frac{QK^T}{\sqrt{d_K}}\right)V \quad (7)$$

where $Attention(Q,K,V)$ is attention score of each vertex v_i at two timestamp t_1 and t_2. After the attention score is obtained, the hidden state of each node can be updated through Eq. (8-9).

$$MultiHead(Q,K,V) = [head_1, ..., head_n]W^O \quad (8)$$

$$head_j = Attention\left(QW_j^Q, KW_j^K, VW_j^V\right) \quad (9)$$

where W_j^Q, W_j^K, W_j^V represents parameter matrices of j-th layer respectively, and $\sqrt{d_K}$ is the vertical dimension of $TGC(X)$, which is used to scale dot products and maintain the stable gradient. $MultiHead(Q,K,V)$ is the output of temporal attention layer. For simplicity, we denote it as $TA(m)$, where $m = TGC(X)$.

Spatial graph convolution: The spatial convolution layer aggregates and transforms the domain information of the nodes to smooth the signal of the central node. We adopt directed diffusion convolution [15] and a self-adaptive adjacency matrix [32] is explored to build the spatial convolution layer as follows.

$$Z = \sum_{k=0}^{K-1} P^{(k)}\xi W_{k1} + \widetilde{A}_{adp}\xi W_{k2} \quad (10)$$

$$\widetilde{A}_{adp} = Softmax\left(ReLU\left(E_1 E_2^T\right)\right) \quad (11)$$

In Eq.(10-11), ξ is input signals which acquired by temporal attention layer $TA(m)$. W_{k1}, W_{k2} correspond to the weight values of matrices, and P denotes the transition matrix that simulates the two-way diffusion. Besides, the self-adaptive adjacency matrix $\widetilde{A}_{adp}$ is calculated with the node embedding of diffusion convolution, which does not require any prior knowledge and is represented by the product of the spatial dependencies of source node embedding E_1 and learnable target node embedding E_2. Also, the $ReLU$ activate function is introduced to eliminate the weak connection relationship between nodes, and $Softmax$ function is used for normalization. Each spatial-temporal layer among the K stacking layers has residual connections. And the output of each layer is concatenated and input to full-connected layers to generate forecast speed.

Objective function: The output of DiffSTG generates P time steps prediction speed sequence $\widehat{X} \in \mathcal{R}^{N\times D\times L}$. We use *Mean Absolute Error* (MAE) as the loss function between predicted values and ground truths as described in Eq.(12).

$$Loss = \frac{1}{|P|}\sum_{L+1}^{L+P} \left|\widehat{X} - X\right| \quad (12)$$

4.4 Dynamic route planning

Traditional route planning methods like Dijkstra is designed with the greedy search for global optimal solution which is time-consuming. Thus, they can't meet the demand of real-time forecasting. In addition, static methods are usually not capable of dynamically changing traffic flows, and consequently, they will not work if one wants to avoid congested roads dynamically. To address these problems, we propose a progressively dynamic route planning algorithm. More specifically, we innovatively propose the multi-step Progressive Dijkstra (PD) and Improved Dynamic Dijkstra (IDD) to handle the time-dependent weighted road network. Meanwhile, we notice that DiffSTG has a cumulative error due to multi-step prediction. Therefore, in consideration of the overall running time with DTT-STG

framework, we use the real-time single-step prediction IDD algorithm instead of the multi-step prediction PD algorithm to reduce the error accumulation. Since the adjacent matrix significantly restricts the speed of Dijkstra, we optimize IDD with the stack data structure, which has a faster execution speed in iterative queries compared to the Dijkstra algorithm using the sparsity of the adjacency matrix, because the speed of Dijkstra's algorithm based on the adjacency matrix is significantly limited by the number of network nodes.

The process of PD and IDD can be concluded as below:

- Utilize Dijkstra to derive the shortest time T for arrival points.
- Find the farthest point that the vehicle can arrive in Δt, if $\Delta t < T$, it means the vehicle can finish the route in a static network. Otherwise, IDD and PD algorithms record the farthest point in Δt.
- Utilize the updated weight to execute Dijkstra network.
- Iterate the above steps in progressive ways until the algorithm reaches the destination within Δt.

As shown in Fig. 5 and Table 2, we compare the different route planning algorithms using the example similar to Fig. 1(c).

We define V as the set of vertices for which the shortest path has been found and indicate U as the set of vertices for which the shortest path has not been found yet. At the initialization step, V contains only the starting point v_1, and the set of U contains all

vertices except for v_1. v_2 is the shortest distance from v_1 to v_2, which will be selected and added to set V. Afterward, the selected vertex will be utilized to update the shortest distance in the iteration procedure until all vertices have been traversed or the query interval Δt is violated. In the example, Dijkstra gets the fastest point v_6 and consumes 122.9 seconds, then progressive Dijkstra will update the weight of the edge. Progressive method detect the surged road speed for $e_{12,13}$ and returns result $(e_{6,8}, e_{8,10}, e_{10,11}, e_{11,13}, e_{13,15})$ and the travel cost is 280.6s and 282.9s according to Table 2, which is different with Dijkstra's output $(e_{6,8}, e_{8,10}, e_{10,12}, e_{12,13}, e_{13,15})$ with travel cost 292.4s.

The process of possessive Dijkstra algorithms (PD and IDD) are summarized in Table 2. Each step represents a vertex being added to the set V. Meanwhile, this vertex's shortest distance and shortest path are computed (denoted as bold values with underscores and path columns, respectively). From the table, we can find that, since step 6 PD and IDD begin to get different travel costs, and the gap increases with the increase of time. The reason is that PD and IDD adopt different traffic flow forecast granularity, IDD can capture the dynamically changed traffic condition with a more micro time granularity. Thus the dynamic progressively traveling time forecasting methods try to exploit the updated speed feature matrix and capture the dynamic complex traffic conditions.

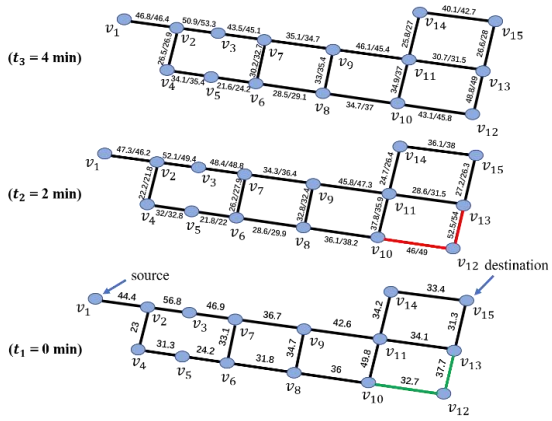


Fig. 5 An example of the shortest path.

$T = t_1$	Path	v_1	v_2	v_3	v_4	v_5	v_6		
Step 0	v_1	**0/0**	∞	∞	∞	∞	∞		
Step 1	v_1, v_2	**0/0**	**44.4/44.4**	∞	∞	∞	∞		
Step 2	v_1, v_2, v_4	**0/0**	**44.4/44.4**	101.2/101.2	**67.4/67.4**	∞	∞		
Step 3	v_1, v_2, v_4, v_5	**0/0**	**44.4/44.4**	101.2/101.2	**67.4/67.4**	**98.7/98.7**	∞		
Step 4	v_1, v_2, v_3	**0/0**	**44.4/44.4**	**101.2/101.2**	**67.4/67.4**	**98.7/98.7**	∞		
Step 5	v_1, v_2, v_4, v_5 v_6	**0/0**	**44.4/44.4**	**101.2/101.2**	**67.4/67.4**	**98.7/98.7**	**122.9/122.9**		
$T = t_2$	Path	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}	v_{14}
Step 6	v_1, v_2, v_4, v_5 v_6, v_7	**149.1/150.8**	151.5/152.8	∞	∞	∞	∞	∞	∞
Step 7	v_1, v_2, v_4, v_5 v_6, v_8	**149.1/150.8**	**151.5/152.8**	184.3/185.2	∞	∞	∞	∞	∞
Step 8	v_1, v_2, v_4, v_5 v_6, v_7, v_9	**149.1/150.8**	**151.5/152.8**	**184.3 / 185.2**	187.6/191	∞	∞	∞	∞
Step 9	v_1, v_2, v_4, v_5 v_6, v_8, v_{10}	**149.1/150.8**	**151.5/152.8**	**184.3 / 185.2**	**187.6/191**	230.1/232.5	∞	∞	∞
Step 10	v_1, v_2, v_4, v_5 v_6, v_8, v_{10}, v_{11}	**149.1/150.8**	**151.5/152.8**	**184.3 / 185.2**	**187.6/191**	**225.4/226.9**	233.6/240	∞	∞
Step 11	v_1, v_2, v_4, v_5 v_6, v_8, v_{10}, v_{12}	**149.1/150.8**	**151.5/152.8**	**184.3 / 185.2**	**187.6/191**	**225.4/226.9**	**233.6/240**	254/256.2	250.1/253.3
Step 12	v_1, v_2, v_4, v_5 v_6, v_8, v_{10}, v_{11} v_{14}	**149.1/150.8**	**151.5/152.8**	**184.3 / 185.2**	**187.6/191**	**225.4/226.9**	**233.6/240**	254/256.2	**250.1/253.3**
Step 13	v_1, v_2, v_4, v_5 v_6, v_8, v_{10}, v_{11} v_{13}	**149.1/150.8**	**151.5/152.8**	**184.3 / 185.2**	**187.6/191**	**225.4/226.9**	**233.6/240**	**254 / 256.2**	**250.1/253.3**
$T = t_3$	Path	v_{15}							
Step 14	v_1, v_2, v_4, v_5 v_6, v_8, v_{10}, v_{11} v_{13}, v_{15}	**280.6/282.9**							

Table 2 Process of IDD and PD algorithm.

5 Experiments

For evaluation, we conduct extensive experiments to discuss the effectiveness of DTT-STG. Section 5.1 introduces two traffic datasets. Section 5.2 briefly illustrates the procedure of data processing and experiment setting. Section 5.3 address the evaluation metrics of each module. Section 5.4 analyze the performance of DTT-STG framework. Sections 5.5 to 5.7 further demonstrate the effectiveness of each module of DTT-STG, including AMM, DiffSTG, and IDD algorithms. Section 5.8 conduct an ablation study to explore the importance of each module.

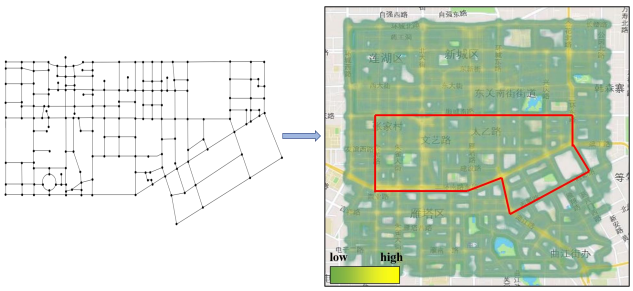


Fig. 6 The processed road structure in Xi'an city.

Dataset	Xi'an	Porto
Latitude range	[108.92, 108.99]	[41.13,41.16]
Longitude range	[34.22, 34.25]	[-8.62, -8.59]
Number of roads	554	314
Number of trajectories	1.22 million	0.89 million

Table 3 The statistics of dataset.

5.1 Data description

To testify DTT-STG, comprehensive experiments are conducted on the Xi'an[1] and Porto[2] datasets. The Xi'an trajectory dataset recorded 2 million floating vehicle trajectories in November 2018, and the Porto dataset recorded 1.7 million floating vehicle trajectories from July 2013 to June 2014. For simplicity, we choose the center area for exploration, including 1.22 and 0.89 million trajectory records in Xi'an and Porto in Fig. 6, respectively. The trajectory is represented as a three-dimensional vector consisting of longitude, latitude, and timestamp. Besides, the topological information is obtained from OpenStreetMap[3], which is formalized as the graph data structure. The summary information of the dataset is shown in Table 3.

5.2 Data preprocessing and experiment setting

As a pre-stage, the result after map matching needs to be preprocessed to ensure effectiveness. First, we removed the point-wise with zero displacements that were incorrectly collected due to temporary stopping. Besides, the speed data higher than 70km/h is also regarded as noise in our experiments with consideration of the speed limit in the urban center.

In addition to noise data, incomplete GPS trajectory data is another challenge, which is caused by GPS or sensor signal malfunction. The traffic volumes between adjacent roads are closely correlated. Therefore, we adopt methods similar to [33]. Based on the temporal correlation, we can take the average of the speed on the adjacent time interval on the road with missing data as the filler value, or use the average of the speed of the road on the same time period on the adjacent date as the filler value. On the spatial level, we can take the average of the speed on the adjacent roads of the missing data as the fill value.

When conducting experiments, all algorithms are implemented in Python and run on an Inter(R) Core (TM) i9-10850K CPU, 3.6GHz, 48GB RAM machine with NVIDIA GeForce RTX 3080Ti.

5.3 Performance measurements

In this section, we give out the performance measurement metrics of each module.

In the map-matching module, If the adjacent values in the road sequence do not match the road network adjacency relationship, that means the original trajectory data has a large number of missing GPS points leading to match failure. And we use the probability of correctly matched trajectories to judge the performance of AMM algorithm.

In the road speed prediction module, we used three evaluation indicators, $RMSE$, MAE, and $MAPE$, which are consistent with literature [16] and [18]. These metrics are defined as Eq.(13)-(15):

In the dynamic path planning module, we believe that the best route under increasingly severe traffic conditions is the one with the least travel time cost under the given confidence threshold as defined at Eq.(1). We hope the predictions returned by the framework minimize the passage time (i.e., improve the compression rate) while being consistent with the real traffic states. Given two paths $route_i$ and $route_j$ that connect the same source and destination, $route_i$ is better than $route_j$ at departure time t, if $route_i$ always gives a higher or equal probability of arrival within a shorter travel time. Based on this intuition, we refer to the *Probabilistic Budget model* according to research [1], and [2] to calculate a given route's confidence. Therefore, we further introduce the travel time $Compression\_rate$ (Eq.(16)) and $Fitting\_rate$ (Eq.(17)) as the evaluation metric of this part. $avg(GT)$ in Eq.(16) and (17) is calculated by all the historical trajectories that passed the querying source and destination point.

$$MAE = \frac{1}{N}\sum_{n=1}^{N}|v_i - \tilde{v}_i| \tag{13}$$

$$MAPE = \frac{100\%}{N}\sum_{n=1}^{N}\frac{|v_i - \tilde{v}_i|}{\tilde{v}_i} \tag{14}$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(v_i - \tilde{v}_i)^2} \tag{15}$$

[1] https://gaia.didichuxing.com
[2] https://www.kaggle.com/crailtap/taxi-trajectory
[3] https://www.openstreetmap.org

$$Compression\_rate = 1 - \frac{route_{opt}.cost - avg(GT)}{avg(GT)}$$

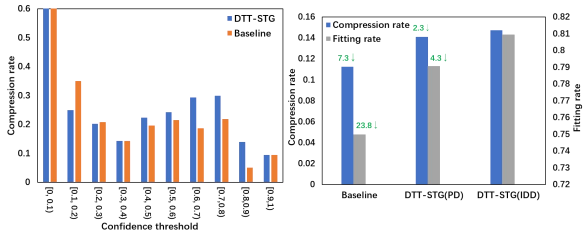
$$if\ route\_opt.cost \le avg(GT) \quad (16)$$

$$Fitting\_rate = \frac{min\ (route_{opt}.cost, avg(GT))}{max\ (route_{opt}.cost, avg(GT))} \quad (17)$$

5.4 Performance of DTT-STG

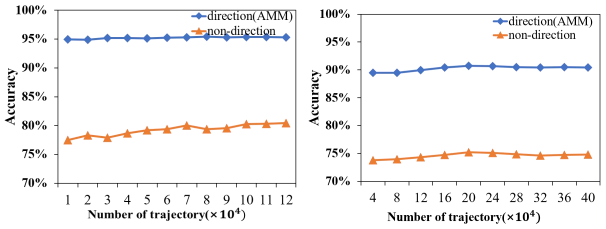
We compare the proposed DTT-STG framework with a baseline model utilizing the Dijkstra algorithm. The comparison metrics are defined in Eq.(16) and Eq.(17). The result is shown in Fig. 7. It is obvious that DTT-STG outperforms the baseline in all circumstances. As shown in Fig. 7(a), with the increase of the confidence threshold, both algorithms' compression rate decreases. This is because when trying to fit the original trajectories, the compression rate inevitably goes down according to the probability distributions of the trajectory dataset. However, with a credible confidence value, our proposed framework can always get a path with a shorter travel time. The reason is that the DTT-STG framework can capture the dynamic changes of the traffic state by the DiffSTG algorithm, which can help choose the real-time optimal route step by step.

Fig. 7(b) compares the models from compression rate and fitting rate according to Eq.(16)-(17). We randomly set up 50 source and destination pairs, and the probability threshold (confidence of a given route) is set to 50%. According to the results, it's obvious that, in all setting metrics, the proposed DTT-STG framework outperforms the baseline, especially the improved DTT-STG(IDD) methods. The compression rate and fitting rate have been increased by up to 7% and 23% respectively, which means that DTT-STG can get the optimal route with the least travel time with a higher possibility. The reason is that DTT-STG employs a DiffSTG traffic flow forecast algorithm which can better capture the dynamically changing traffic status, and DTT-STG further utilizes the progressive dynamic routing methods which can avoid congested roads dynamically. Notice that, although Dijkstra sometimes returns a route with smaller cost, the route is impracticable considering the morning and evening rush hours which always along with the dramatically increasing traffic flow and result in longer traveling time. In other words, the route planned by Dijkstra cannot satisfy the confidence (probability) threshold. And in Fig. 7(b) we only consider the routes that satisfy the confidence threshold.



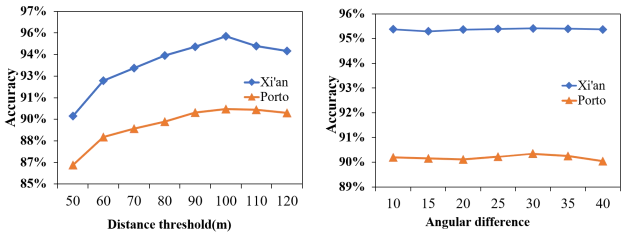
(a) Confidence threshold and Compression rate. (b) Compression and Fitting rate.

Fig. 7 Compression of models based on different metrics.



(a) Xi'an (b) Porto

Fig. 8 The change trend of map-matching accuracy and with increasing of trajectory dataset's size.



(a) Effect of distance threshold (b) Effect of direction angle

Fig. 9 Effects of parameters on the algorithms accuracy

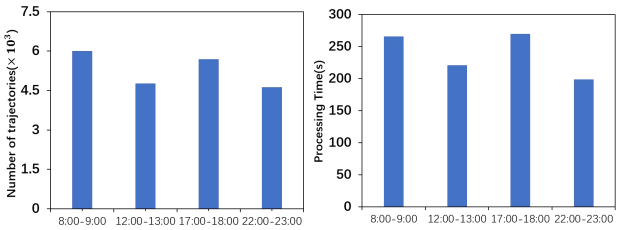


Fig.10 Update frequency and efficiency of AMM

5.5 Effectiveness and efficiency of AMM

Map-matching is an important component of DTT-STG, which preprocess the trajectory. In DTT-STG, we leverage the non-direction based method by introducing angle information and innovatively propose the Angle-based map-matching algorithm (AMM). According to Fig. 8, the precision of AMM is about 95.3% and 90.3% in Porto and Xi'an datasets respectively,

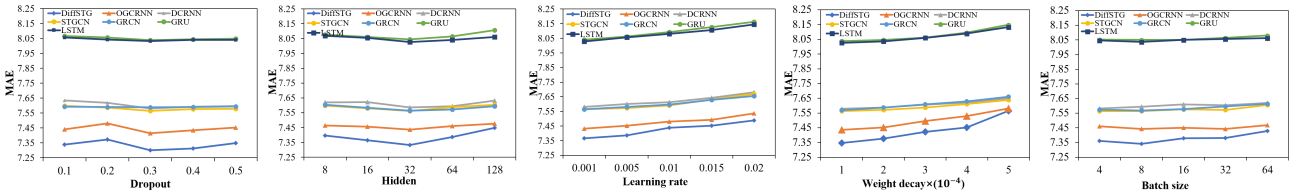


Fig. 11 Parameter sensitivity with Xi'an dataset.

Dataset	Xi'an traffic dataset at 5/10/15/20-minute interval		
Model	MAE	MAPE(%)	RMSE
LSTM	8.03/8.03/8.03/8.03	5.19/5.19/5.19/5.21	10.92/10.92/10.92/10.92
GRU	8.04/8.04/8.06/8.04	5.20/5.21/5.21/5.21	10.92/10.92/10.92/10.92
GRCN	7.58/7.58/7.57/7.57	4.86/4.87/4.87/4.87	10.43/10.43/10.43/10.43
STGCN	7.57/7.57/7.57/7.58	4.82/4.81/4.77/4.82	10.47/10.47/10.46/10.49
DCRNN	7.58/7.59/7.59/7.59	4.83/4.82/4.84/4.85	10.48/10.49/10.49/10.53
OGCRNN	7.44/7.43/7.44/7.45	4.78/4.79/4.75/4.79	10.36/10.35/10.35/10.36
DiffSTG(ours)	**7.33/7.33/7.33/7.34**	**4.67/4.71/4.68/4.67**	**10.21/10.21/10.20/10.21**
Dataset	Porto traffic dataset at 5/10/15/20-minute interval		
Model	MAE	MAPE(%)	RMSE
LSTM	5.54/5.67/5.74/5.74	4.10/4.17/4.19/4.34	7.62/7.88/7.97/8.00
GRU	5.50/5.57/5.70/5.81	4.15/4.09/4.17/4.33	7.58/7.84/7.95/7.99
GRCN	5.28/5.33/5.42/5.51	3.78/3.85/3.92/4.08	7.42/7.49/7.65/7.70
STGCN	5.15/5.18/5.21/5.25	3.72/3.78/3.83/3.95	7.22/7.25/7.28/7.32
DCRNN	5.18/5.22/5.30/5.35	3.73/3.79/3.84/3.97	7.27/7.30/7.39/7.43
OGCRNN	5.08/5.13/5.17/5.21	3.64/3.70/3.76/3.84	7.07/7.13/7.20/7.24
DiffSTG(ours)	**5.00/5.04/5.06/5.10**	**3.59/3.64/3.69/3.71**	**6.96/7.02/7.09/7.15**

Table 4 Performance comparison different approaches and intervals.

which is significantly higher than non-direction based methods, and the average accuracy improvement is about 17%. Most trajectories are correctly mapped to the road network. We further discuss the selection of distance threshold and direction angle in Fig. 9. As the value of the distance parameter increases, the overall accuracy of AMM increases, and our model is able to maintain stability in terms of angle changes.

In particular, we consider the efficiency of AMM and its update frequency within a given Δt. It determines how often we need to execute the map-matching algorithm to maintain the framework running. The results are displayed in Fig. 10. With ten processes in parallel, AMM is capable of processing the trajectory datagenerated in one hour considering the Xi'an dataset. And the execution time is about 200 to 300 seconds, which can satisfy the minimum querying time interval ($\Delta t = 5\text{min}$) of the proposed dynamic routing algorithms.

5.6 Comparison of different road speed forecast methods

After map-matching, the road speed should be estimated for dynamic traveling time forecast, which is the second core module in DTT-STG. Specifically, we design a novel spatial-temporal graph convolution layer with the attention and diffusion mechanism called DiffSTG to estimate the road speed dynamically. Establishing a clear superiority of our proposal, we compare DiffSTG with some competitive prediction models in $RMSE$, MAE, and $MAPE$ evaluation terms. The definition of these evaluation terms has been introduced in Section 5.2.

The datasets are split with 70% for training, 10% for validation, and 20% for testing in chronological order. Besides, all competitive models have been trained 50 epochs by Adam optimizer with batch size 4, learning rate 0.001, and weight decay value of 0.0001. To guarantee the fairness, we utilize the same hyperparameter settings, input features, and partition methods when implementing the experiment, which keeps consistency with [17], [35]. Grid search is adopted to find out the optimal hyperparameters, and the experiment result is shown in Fig. 11. It's obvious that the proposed DiffSTG module outperforms other algorithms in all settings.

LSTM: A recurrent neural network constructed with forget gate, update gate, and output gate with gate structure, which enables forgetting or remembering historical information.

GRU: A recurrent neural network constructed with update gate and output gate.

GRCN [34]: A model that uses RNNs to query dynamic patterns while using graph structure and CNNs to identify the spatial structure for processing structured sequences (e.g., spatio-temporal sequences)

DCRNN [15]: A diffusion convolution recurrent neural network, which combines graph convolution networks with recurrent neural networks in an encoder-decoder manner.

STGCN [35]: A spatial-temporal graph neural network which can capture spatial-temporal features dynamically. STGCN block composes of Graph convolution layer and gated temporal convolution layer (GRU) to extract the most essential spatial and

temporal dependencies respectively.

OGCRNN [36]: A spatial and temporal neural network is constructed for traffic prediction based on GCN and GRU networks, which combines the graph updating strategy for constructing optimized graph matrix.

Comparison	Wilcoxon signed-rank test $\alpha = 0.05$	Friedman test
DiffSTG vs. LSTM	1.6×10^{-5}	1.6×10^{-9}
DiffSTG vs. GRU	1.6×10^{-5}	
DiffSTG vs. GRCN	6.5×10^{-3}	
DiffSTG vs. STGCN	4.2×10^{-3}	
DiffSTG vs. DCRNN	4.7×10^{-3}	
DiffSTG vs. OGCRNN	2.7×10^{-5}	

Table 5 Results of *p*-value used Wilcoxon signed-rank test and Friedman test.

Since the highlight of this paper is dynamic traveling time forecast, we conduct the DiffSTG algorithm in different time intervals (Δt), including 5, 10, 15, and 20 minutes as shown in Table 4. According to the characteristics of the model, RNN-based models (LSTM and GRU) consider the time-series information, while GCN-based models (GRCN, STGCN, DCRNN, OGCRNN, and DiffSTG) further supplement the spatial topological information. Thus, the prediction error of GCN-based models is relatively lower than that of RNN-based models. Besides, the experiments show strong evidence that DiffSTG further outperforms other models in all metrics with the attention and diffusion mechanism. Moreover, DiffSTG achieves about 5% average precision improvement.

Furthermore, the significant test is run for the evaluation result in Table 5. The Wilcoxon signed-rank test is conducted to test whether the distribution of the value $Error_{DiffSTG} - Error_{other}$ is symmetric about zero, where $Error_{DiffSTG}$, $Error_{other}$ denotes the estimated error of DiffSTG and other competitive methods respectively. In other words, the test value will be extremely lower when $Error_{DiffSTG}$ is always lower than $Error_{other}$. Besides, we run the Friedman tests to evaluate whether the outputs of different models share the same distribution. Observing the values in Table 5, DiffSTG always performs better than other off-the-shelf models since the significance level *p* is far less than 0.05 (reject *H0*).

5.7 Comparison of different route planning methods

Based on the result of map-matching and road speed estimation, DTT-STG explores several dynamic route planning methods for travel time estimation. Taking account of the changeable traffic congestion states, we propose the Improved Dynamic Dijkstra (IDD) algorithm. To verify the superiority of IDD, we randomly select 200 trajectory data for each period, i.e., 8:00-9:00, 12:00-13:00, 17:00-18:00, and 22:00-23:00 on November 30. Note that, we derive the road network weight (i.e., travel time) according to the speed value computed by DiffSTG and the distance of each road.

Fig.12 compares four different route planning algorithms, i.e., Floyd, Dijkstra, and our proposed IDD and PD algorithm. The former two algorithms are static planning methods that only take consideration of the static road network information on the departure time, while DTT-STG with IDD and PD model derives the optimal route based on the time-varying road network graph. And the probability threshold (confidence of a given route) is set to 50%.

The traveling time of the route planned by DTT-STG framework is shorter than that of the static algorithm and ground truth in all time periods, and the compression rate is about 10% compared to the real trajectory. Fig.12 also shows the traveling time saved by IDD compared to the static route planning algorithm and the original trajectory, DTT-STG framework can save traveling time significantly, no matter which route planning algorithm to choose (either IDD or PD). Moreover, IDD and PD algorithm can get a traveling compression rate up to 11.9% and 10.9% respectively. The proposed algorithms' superiority is obvious especially during the morning and evening peak. In summary, DTT-STG framework can help us save plenty of commuting time.

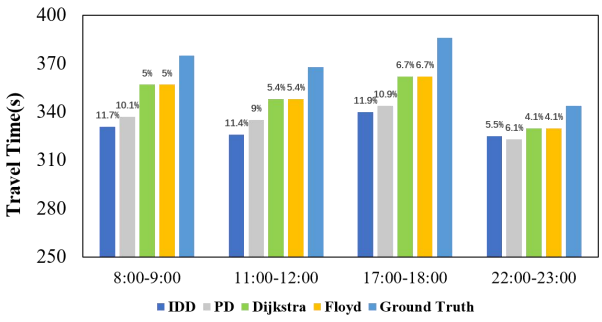


Fig. 12 Parameter sensitivity with Xi'an dataset.

Framework	MAE	MAPE(%)	RMSE
DTT-STG w/o AMM	84.37	40.67	118.51
DTT-STG w/o DiffSTG	96.84	51.88	134.56
DTT-STG w/o IDD	78.21	37.35	108.04
DTT-STG	**75.06**	**35.78**	**104.21**

Table 6 Ablation analysis of each module.

5.8 Ablation study

In our proposed DTT-STG framework, each module is an indispensable part. To investigate the importance of each module to the prediction result, we conduct the ablation experiments in Table 6. Specifically speaking, we do not set the confidence limitation for the routes which given by all frameworks. For each ablation framework, the default map-matching algorithm is a distance-based method, and the road travel speed is calculated by a statistical method based on historical average traffic flow at the current moment, and Dijkstra is the default route planning algorithm. In this section, we randomly set up 50 source and destination pairs by comparing the travel time returned by the DTT-STG framework and the real average travel time of the actual trajectory between departure and destination. It's obvious that the road speed prediction module is the core component of the framework since the performance of DTT-STG degrades dramatically with no DiffSTG module. Map-matching (AMM) and dynamic route planning (IDD) also contribute to the optimization of the framework since the error rate increases significantly without these two modules.

6 Conclusion

With the prevalence of GPS navigation systems, route plan with dynamic traveling time forecasting in rapidly changing traffic conditions has become a new research highlight. After a careful investigation, it is found that very limited work considers the dynamics and proximity of traffic flows, which is difficult to dynamically plan the shortest travel time route according to real-time traffic information. To resolve the defects, this paper is dedicated to designing a Dynamic Traveling Time forecasting framework based on Spatial and Temporal Graph convolution (DTT-STG) with full consideration of dynamics and proximity. In DTT-STG, the Angle-based map-matching (AMM) module aligns a sequence of observed positions on a trajectory to a digital map and introduces the direction information of the moving vehicle. Then, the Diffusion Spatial-Temporal Graph convolutional network (DiffSTG) is exploited to estimate the road speed on each road segment, which takes care of the attention of spatial diffusion and temporal convolution. Finally, the Improved Dynamic Dijkstra (IDD) is introduced dynamically plan the shortest travel time route in continuously changing traffic conditions.

The experimental results indicate that DTT-STG has achieved surprising achievements with the integration of AMM, DiffSTG, and IDD components in the task of dynamic traveling forecast. Firstly, AMM algorithm has an excellent performance in processing trajectories in Xi'an and Porto datasets, which have 95.3% and 90.3% accuracy and get 17% higher than the non-direction algorithm, respectively. And then, DiffSTG utilizes the actual road network to construct spatial relations, which achieves 5% average precision improvement in road speed prediction compared with state-of-the-art models. Furthermore, the route planned by IDD algorithm can save around 10% traveling time compared with ground truth, which can avoid congested roads and make more intelligent decisions, especially in peak hour traffic congestion state.

References

1. Pedersen S A, Yang B, Jensen C S. Fast stochastic routing under time-varying uncertainty. VLDB J., 2020, 29(4): 819–839
2. Guo C, Yang B, Hu J, Jensen C S. Learning to route with sparsetrajectory sets. In: 34th IEEE International Conference on Data Engineering, ICDE 2018, Paris, France, April 16-19. 2018, 1073–1084
3. Zheng J, Gao Z, Ma J, Shen J, Zhang K. Deep graph convolutional networks for accurate automatic road network selection. ISPRS International Journal of Geo-Information, 2021, 10(11): 768
4. Chen F, Qi Y, Wang J, et al. Temporal Metrics Based Aggregated Graph Convolution Network for traffic forecasting[J]. Neurocomputing, 2023, 556: 126662.
5. Liu H, Jin C, Zhou A. Popular route planning with travel cost estimation from trajectories. Frontiers of Computer Science, 2020, 14: 191–207
6. Liu X, Zhang D, Yan H, Cui Y, Chen L. A new algorithm of the best path selection based on machine learning. IEEE Access, 2019, 7: 126913–126928
7. Zhu D D, Sun J Q. A new algorithm based on dijkstra for vehicle path planning considering intersection attribute. IEEE Access, 2021, 9: 19761–19775
8. Li P, Wang X, Gao H, Xu X, Iqbal M, Dahal K. A dynamic and scalable user-centric route planning algorithm based on polychromatic sets theory. IEEE Transactions on Intelligent Transportation Systems, 2021, 23(3): 2762–2772
9. Ning Z, Sun S, Zhou M, Hu X, Wang X, Guo L, Hu B, Kwok R Y. Online scheduling and route planning for shared buses in urban traffic networks. IEEE Transactions on Intelligent Transportation Systems, 2021, 23(4): 3430–3444
10. Yang B, Yuan L, Yan J, Ding Z, Cai Z, Guo L. A novel heuristic

method for emergency path planning based on dynamic spatialtemporal characteristics map. In: Journal of Physics: Conference Series. 2021, 012005

11. Sun N, Shi H, Han G, Wang B, Shu L. Dynamic path planning algorithms with load balancing based on data prediction for smart transportation systems. IEEE Access, 2020, 8: 15907–15922
12. Liu Z, Li Z, Wu K, Li M. Urban traffic prediction from mobility data using deep learning. IEEE network, 2018, 32(4): 40–46
13. Liu Y, Zheng H, Feng X, Chen Z. Short-term traffic flow prediction with conv-lstm. In: 2017 9th International Conference on Wireless Communications and Signal Processing (WCSP). 2017, 1–6
14. Wu Z, Pan S, Chen F, Long G, Zhang C, Philip S Y. A comprehensive survey on graph neural networks. IEEE transactions on neural networks and learning systems, 2020, 32(1): 4–24
15. Li Y, Yu R, Shahabi C, Liu Y. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. arXiv preprint arXiv:1707.01926, 2017
16. Guo K, Hu Y, Sun Y, Qian S, Gao J, Yin B. Hierarchical graph convolution networks for traffic forecasting. In: Proceedings of the AAAI Conference on Artificial Intelligence. 2021, 151–159
17. Zhao L, Song Y, Zhang C, Liu Y, Wang P, Lin T, Deng M, Li H. Tgcn: A temporal graph convolutional network for traffic prediction. IEEE Transactions on Intelligent Transportation Systems, 2019, 21(9): 3848–3858
18. Cai L, Yan B, Mai G, Janowicz K, Zhu R. Transgcn: Coupling transformation assumptions with graph convolutional networks for link prediction. In: Proceedings of the 10th International Conference on Knowledge Capture. 2019, 131–138
19. Dai G, Hu X, Ge Y, Ning Z, Liu Y. Attention based simplified deep residual network for citywide crowd flows prediction. Frontiers of Computer Science, 2021, 15: 1–12
20. Ebendt R, Sohr A, Touko Tcheumadjeu L C, Wagner P. Utilizing historical and current travel times based on floating car data for management of an express truck fleet. 2010
21. Brakatsoulas S, Pfoser D, Salas R, Wenk C. On map-matching vehicle tracking data. In: Proceedings of the 31st international conference on Very large data bases. 2005, 853–864
22. Newson P, Krumm J. Hidden markov map matching through noise and sparseness. In: Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems. 2009, 336–343
23. Lou Y, Zhang C, Zheng Y, Xie X, Wang W, Huang Y. Mapmatching for low-sampling-rate gps trajectories. In: Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems. 2009, 352–361
24. Chen B Y, Yuan H, Li Q, Lam W H, Shaw S L, Yan K. Map-matching algorithm for large-scale low-frequency floating car data. International Journal of Geographical Information Science, 2014, 28(1): 22–38
25. Dijkstra E W. A note on two problems in connexion with graphs. Numerische Mathematik, 1959, 1(1): 269–271
26. Yalin C, Liyang Z, Longbiao Z, Xiaojiang S, Heng W. Research on path planing of parking system based on the improved dijkstra algorithm. Modern Manufacturing Engineering, 2017, 443(8): 63–67
27. Chabini I, Lan S. Adaptations of the a* algorithm for the computation of fastest paths in deterministic discrete-time dynamic networks. IEEE Transactions on Intelligent Transportation Systems, 2002, 3(1): 60–74
28. Liu Y, Wu H. Path planning based on theoretical shortest distance variable weight a* algorithm. Comput. Meas. Control, 2018, 26(4): 15 175–178
29. Zuo X, Shen W. Improved algorithm about muti-shortest path problem based on floyd algorithm. Computer science, 2017, 44(5): 232–267
30. Oord A v d, Dieleman S, Zen H, Simonyan K, Vinyals O, Graves A, Kalchbrenner N, Senior A, Kavukcuoglu K. Wavenet: A generative model for raw audio. arXiv preprint arXiv:1609.03499, 2016 30.
31. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez A N, Kaiser Ł, Polosukhin I. Attention is all you need. Advances in neural information processing systems, 2017, 30
32. Wu Z, Pan S, Long G, Jiang J, Zhang C. Graph wavenet for deep spatial-temporal graph modeling. arXiv preprint arXiv:1906.00121, 2019
33. Guo F, Zhang D, Dong Y, Guo Z. Urban link travel speed dataset from a megacity road network. Scientific data, 2019, 6(1): 61
34. Seo Y, Defferrard M, Vandergheynst P, Bresson X. Structured sequence modeling with graph convolutional recurrent networks. In: International Conference on Neural Information Processing. 2018, 362–373
35. Yu B, Yin H, Zhu Z. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. arXiv preprint arXiv:1709.04875, 2017
36. Guo K, Hu Y, Qian Z, Liu H, Zhang K, Sun Y, Gao J, Yin B. Optimized graph convolution recurrent neural network for traffic prediction. IEEE Transactions on Intelligent Transportation Systems, 2020, 22(2): 1138–1149