Pattern Recognition 158 (2025) 110978

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Trajectory-User Linking via Multi-Scale Graph Attention Network

Yujie Li [a,b], Tao Sun [a], Zezhi Shao [a,b], Yiqiang Zhen [c], Yongjun Xu [a,b], Fei Wang [a,b,*]

[a] Institute of Computing Technology, Chinese Academy of Sciences, Beijing, 100190, China
[b] University of Chinese Academy of Sciences, Beijing, 100049, China
[c] DFH Satellite Co., Ltd., Beijing, 100094, China

ARTICLE INFO

Keywords:
Trajectory-user linking
Graph neural network
Trajectory classification
Spatio-temporal data mining
Check-in data

ABSTRACT

Trajectory-User Linking (TUL) aims to link anonymous trajectories to their owners, which is considered an essential task in discovering human mobility patterns. Although existing TUL studies have shown promising results, they still have specific defects in the perception of spatio-temporal properties of trajectories, which manifested in the following three problems: missing context of the original trajectory, ignorance of spatial information, and high computational complexity. To address those issues, we revisit the characteristics of the trajectory and propose a novel model called TULMGAT (TUL via Multi-Scale Graph Attention Network) based on masked self-attention graph neural networks. Specifically, TULMGAT consists of four components: construction of check-in oriented graphs, node embedding, trajectory embedding, and trajectory user linking. Sufficient experiments on two publicly available datasets have shown that TULMGAT is the state-of-the-art model in task TUL compared to the baselines with an improvement of about 8% in accuracy and only a quarter of the fastest baseline in runtime. Furthermore, model validity experiments have verified the role of each module.

1. Introduction

Benefiting from the proliferation of GPS-based devices and the rapid development of mobile applications in recent years, large amounts of trajectories, which are converted from check-in records, have been recorded. Those trajectories contain spatio-temporal information about human individual mobility patterns. It has led to the emergence of location-based social networking(LBSN) technology, which is regarded as an essential bridge between the virtual web and the real world. As check-in records containing spatio-temporal features is one of the most common information generated by personal daily routines, LBSN has many practical applications, such as functional region discovery [1], trajectory anomaly detection [2], and so on. Trajectories generated by user movement are special spatio-temporal data converted from check-in records in LBSN after initial characterization such as discretization and embedding [3].

Trajectory-user linking [4] is a critical task in trajectory analysis, receiving much attention in recent studies. It aims to link anonymous trajectories to the users who generated them by learning from known user trajectories. Due to increased population and refined urban functional areas, the transportation environment of modern cities tends to be complicated [5]. Applications such as social networks and navigation tend to use trajectories to optimize user-related services. However, for the protection of personal information [6], third-party service providers often cannot get trajectories containing users' labels due to privacy agreements, so trajectory user linking has a practical application in the field of intelligent transportation and is a fundamental task for analyzing human mobility and individual movement patterns.

There are two types of solutions for trajectory user linking: traditional trajectory similarity-based studies and deep learning-based models. Traditional methods often use statistical machine learning to analyze spatial properties of trajectory points, such as Deep Bayesian Networks (DBN), Hidden Markov Models (HMM), and Longest Common Sub-Sequence (LCSS) [7]. These methods are considered feasible for trajectory user linking. However, the available research at TUL verifies that the traditional schemes struggle with linear nonseparability and noise in large trajectory datasets, which are the prevalent difficulties faced by the similarity computation of pairwise points-matching [7].

For those reasons, deep learning solutions have been proposed and are gradually dominating the topic. Trajectory user linking is a spatio-temporal data problem formally defined by [4] in 2017, and TULER was proposed to establish the research foundation for the subsequent research of TUL based on AI-based methods [8]. Inspired by Word2vec [9] in NLP, TULER treats POIs [10] containing spatial

* Corresponding author at: Institute of Computing Technology, Chinese Academy of Sciences, Beijing, 100190, China.
E-mail address: wangfei@ict.ac.cn (F. Wang).

https://doi.org/10.1016/j.patcog.2024.110978
Received 25 November 2023; Received in revised form 2 September 2024; Accepted 2 September 2024
Available online 3 September 2024

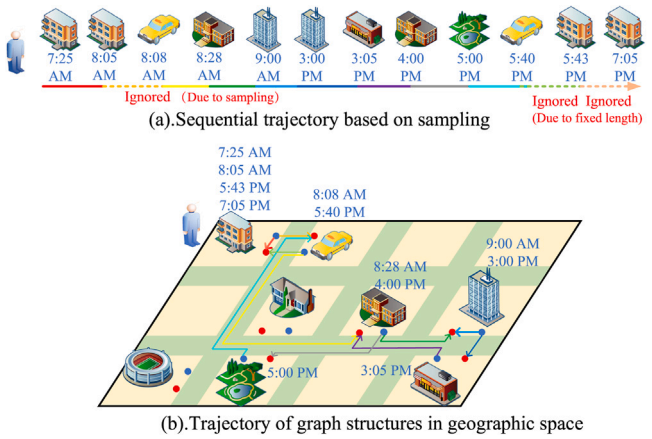

Fig. 1. The structure of trajectory.

information as words to embed POIs from the two-dimensional geographic space into high-dimensional space and continues to mine temporal dependencies and spatial properties in trajectories by RNN-based models.

Following TULER, TULVAE [11], STULIG [12] and others utilize semi-supervised generative networks [13] to assist in classification tasks. TULAR [14] addresses the issue of information loss or feature redundancy in previous models due to the sampling of a fixed number of POIs by introducing the Trajectory Semantic Vector (TSV) module to embed trajectories of arbitrary length. However, these works treat trajectories as a variant of text or a type of time series, as shown in Fig. 1(a), rather than as spatio-temporal data. GNNTUL [15] is the first model to consider a trajectory as a graph, similar to Fig. 1(b), and explores the spatial features of location information in trajectories, TULRN [16] is a variant of GNNTUL by improving the neighbor sampling scheme of GraphSAGE [17].

Despite the significant contributions of the works mentioned above, the following problems limit their performances: (1). **Missing context of the original trajectory**; (2). **Ignorance of spatial information**; (3). **High computational complexity**. Next, we will elaborate on these common issues in existing work in conjunction with Table 1:

(1). **Missing context of the original trajectory**: This issue arises due to the limitations of the preprocessing strategy or the pre-defined modeling strategy. In the models we discuss except TULAR [14], other work faces the problem of missing original trajectory contexts. Since users actively report check-in records, the number of check-ins over time is random and the intervals between check-ins vary, leading to the inconsistent trajectories lengths.

On one hand, **RNN-based methods** like TULER [4] treat trajectories as sequences shown in Fig. 1, but cannot handle variable-length trajectories. Thus, they need to randomly sample to discard or repeat some check-ins, or directly truncate long trajectories and pad short trajectories to obtain a fixed-length trajectory for processing, which explains why some check-ins of the long trajectory in Fig. 1(a) have been ignored.

On the other hand, existing **GNN-based methods**, such as GNNTUL, not only intercept a fixed number of nodes in the trajectory to construct the graph, but also apply GraphSAGE to sample a fixed number of neighboring nodes for aggregation. Even though TULRN proposes to utilize Renyi entropy to compute nodes weights to improve sampling, it still inevitably loses the original trajectory information. In summary, RNN-based models are limited by fixed-length inputs, and GraphSAGE-based models tend to be limited by aggregation algorithms, resulting in the inability to handle trajectories of arbitrary length at the model level, and their sampling results in missing context for the original trajectories at the data level.

Table 1
Comparison of TULMGAT with others methods in motivation.

Model	Context of the trajectory	Spatial information	Computational complexity
TULER	Incomplete	None	Medium
TULVAE	Incomplete	None	High
TULAR	Complete	None	Medium
STULIG	Incomplete	None	High
GNNTUL	Incomplete	True	Medium
TULRN	Incomplete	True	Medium
The desired model (TULMGAT)	Complete	True	Low

(2). **Ignorance of spatial information**: As previously mentioned, most TUL methods, except GNNTUL and TULRN, treat trajectories as text variants or sequence data in Fig. 1(a), and cascade location embeddings to mine human mobility patterns, which ignores the spatial information of trajectories in Fig. 1(b), resulting in ignoring the connections in geospatial space, with the deeper semantics of geographic location.

(3). **High computational complexity**: TUL methods like TULVAE [11] and STULIG [12] introduce generative tasks to mine potential trajectory distributions, resulting in unaffordable training costs due to the larger number of parameters in trajectory generation tasks compared to classification tasks. Additionally, RNN-based methods, due to tandem location embedding, and GNNTUL [15] and TULRN [16], which aggregate neighboring nodes in a cascade manner, lack advantages in parallel computation. A detailed analysis of computational complexity is presented in Table 6 of Section 5.2.3.

Among the three problems mentioned above, the missing context of the original trajectory and the ignorance of spatial information result in incomplete trajectory information, reducing the accuracy of trajectory-user linking. High time complexity affects training speed and quick deployment.

We believe that existing work suffers from these problems because they do not treat trajectories as complete and ideal spatio-temporal data as shown in Fig. 1(b), and we propose a trajectory user-linking model that retains all original trajectory information while efficiently mining spatio-temporal features. Therefore, the model needs to satisfy the following requirements compared to previous works:

(1). Context: Allows the information in the original trajectory of any length to be taken into account.

(2). Spatial Information: Be able to fully exploit the spatial properties of the trajectories.

(3). Computational Complexity: Be able to compute the interactions of spatial information in parallel.

(4). Robustness: Additional issues are required in the trajectory task, which we explain in detail below:

Our interpretation of robustness is that trajectory user linking will face a more severe class-imbalance problem compared to general classification tasks. Since users' check-in behavior is based entirely on passive acquisition, the number of collected check-in records and trajectories varies from user to user, as confirmed by Fig. 2. Gowalla, Brightkite, and Foursquare are well-known social platforms open-sourcing check-in record datasets. The differences between users in terms of number of check-ins and trajectories, and length of trajectories are huge. One notable problem is the varying length of trajectories, leading to extreme differences in the raw data received by the model, similar to noise and distortion issues in image classification. The realistic requirement that the model should not fail in classifying trajectories of varying lengths from different users poses a significant challenge to its robustness, which we need to alleviate.

Having observed the drawbacks of existing work on the TUL task and determined the direction for improvement, we propose a novel **T**rajectory-**U**ser **L**inking model based on the **M**ulti-scale **G**raph

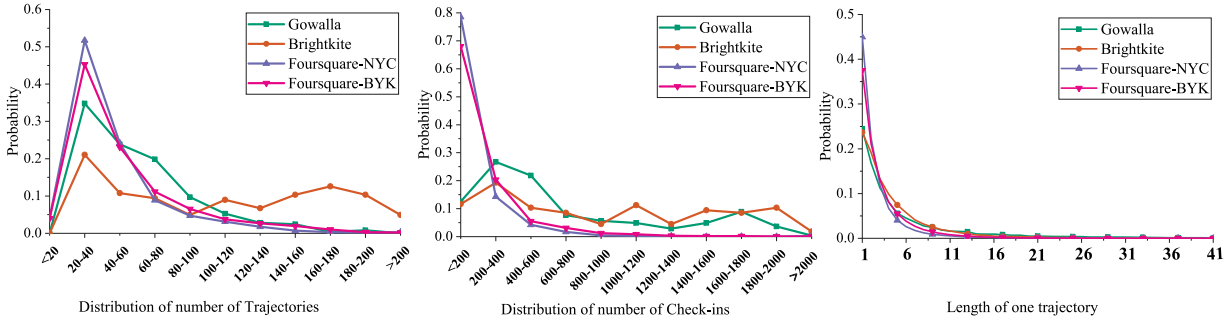

Fig. 2. Imbalance of samples in terms of the number of trajectories/check-ins, and trajectory length.

Attention network, namely TULMGAT, which aims to address the aforementioned problems and enhance robustness.

TULMGAT consists of the following four components.

(1) **Construction of check-in oriented graphs** *COGs*: We first divide the geographic space into grid cells, then process a single trajectory into a multi-scale collection of trajectories based on different sampling rates, and finally construct multiple graphs with the grid representation of trajectories and pre-defined rules.

(2) **Node embedding**: Masked multi-head self-attention graph neural network [18] is employed to learn the representation of an arbitrary number of nodes and Global Average Pooling is adopted to compress these node vectors to a uniform dimension.

(3) **Trajectory embedding**: Unlike RNN-based methods that use temporal models to discover temporal dependencies between nodes, the BiLSTM we apply is designed to fuse information between scales.

(4) **Linking**: A simple multi-layer perceptron (MLP) and softmax function-based module is proposed to link trajectories to their owners.

In addition, we emphasize again that we simultaneously solve the three main problems above in trajectory-user linking and achieve excellent performance through the novel model we propose. This accurate and expeditious approach can be quickly applied to real-world scenarios requiring trajectory-user linking, such as optimizing social network services associated with anonymous users and improving classification and anomaly analysis performance. Effectively linking anonymous trajectories to known users is considered crucial for mining human mobility patterns and plays an important role in social networks, transportation, urban planning, social security, and other fields.

We summarize the main four contributions of our work as follows.

- We propose a novel model called TULMGAT to accomplish the TUL task more effectively and efficiently by simultaneously solving the above problems (missing context of the original trajectory, ignorance of spatial information, high time complexity).
- To better represent the spatial information in trajectories, we construct check-in oriented graphs *COGs*. To improve efficiency and effectively mine spatial interactions in trajectories, we employ a masked multi-head self-attention graph neural network with parallel computing capabilities. To fully retain trajectory information, we consider global average pooling to allow trajectories of arbitrary length to be explored rather than sampling the trajectories.
- To further improve the effectiveness and robustness of TULMGAT, we sample the trajectories with different sampling rates and design the model to capture the human mobility patterns at multiple scales based on retaining all valid information in the trajectories.
- The extensive experiments conducted on two publicly available datasets, Gowalla and Brightkite, demonstrate that TULMGAT outperforms all compared approaches regarding effectiveness and efficiency. Moreover, sufficient additional experiments were conducted to verify the effectiveness and robustness of the model.

2. Related work

2.1. Trajectory-user linking

Trajectory-user linking is a classification or recognition task that links anonymous trajectories to known users who generated them by mining spatio-temporal properties. TULER [4] is the first model to formulate and investigate this task. Inspired by word embedding in NLP, TULER regards geographic information in a trajectory as words and the entire trajectory as a sentence, thus embedding two-dimensional spatial information into high-dimensional vectors, which allows deep learning methods to learn underlying information in the trajectory. TULER initially proposed three variant models based on RNN methods, namely TULER-LSTM, TULER-GRU, and BiTULER.

Following this work, TULVAE [11] exploits variational inference to mine latent trajectory distributions and applies semi-supervised learning to include unlabeled data in training. Both STULIG [12] and TGAN [19] utilize generative models to enhance the original training set, with STULIG employing a Variational AutoEncoder (VAE) and TGAN using a generative adversarial network. DeepTUL [20] designs an attention module based on historical trajectories to obtain the context for understanding human mobility patterns, concatenating this context with the trajectory vector learned by an RNN-based model. TULAR [14] is the first model that addresses the loss of context in previous models due to fixed-length trajectory sampling by developing the Trajectory Semantic Vector (TSV) module to embed arbitrary-length trajectories and capture hidden mobility features.

While these studies utilize sequential information in trajectories, they do not effectively use spatial information. GNNTUL [15] tries to exploit the structural features of trajectories and mines complex intrinsic geospatial information by sampling a fixed number of sources nodes and neighboring nodes and utilizing GraphSAGE [17] to learn their features. TULRN [16] enhances graph construction with priori road networks and uses Renyi entropy to compute spatial node weights for sampling in GraphSAGE.

In addition, there are several programs in trajectory user linking. Introducing unique additional information is a feasible idea. For example, MTUL [21] adds POI category labels as spatial dimensions. Furthermore, some reinforcement learning methods are applied to TUL. For instance, MainTUL [22] applies distillation learning networks for reinforcement of trajectory representation, and TULMAL [23] employs generative adversarial learning to optimize trajectories representation through a multi-task learning framework with semi-supervised learning. However, while additional information improves trajectory representation, it lacks flexibility and generalizability with new datasets. Moreover, reinforcement learning schemes do not specifically address trajectory characteristics and can be applied to any model, so they are not used as baselines for comparative experiments.

Table 2
Description of key notations.

Notations	Descriptions
u, r, g, τ	User, check-in record, gird cell, trajectory
U, T	Users set, trajectories set
l	Location information of a check-in record
t	Timestamp of a check-in record
$G(\tau)$	Grid representation of trajectory τ
$\mathbf{e}_\tau$	Embedding of the trajectory τ of GAT output
$\mathbf{e}'_\tau$	Embedding of the trajectory τ of BiLSTM output
$COGs$	Graphs generated by corresponding trajectories
v_i	Node i in graph COG
e_{ij}	Edge between node v_i and node v_j
$\mathbf{h}_i$	Representation of the node v_i
$\mathbf{h}'_i$	Representation of the node v_i of GAT output
N_G	Number of $COGs$
N_i	Number of nodes in COG_i
N_U	Number of users
D_v	Embedding dimension of node v
D_G	Embedding dimension of $\mathbf{h}'_i$
D_R	Embedding dimension of $\mathbf{e}'_\tau$

2.2. Graph neural networks

The connections between different items are widely available in reality, expressed as social networks and knowledge graphs [24]. Research related to graphs has attracted increasing attention in the last decade. Benefiting from the development of deep learning networks and graph theory, we can learn nodes features through neural networks.

GCN [25] is one of the earliest graph neural networks, which generalizes convolutional methods from traditional image to graph. The core idea of GCN lies in the eigen decomposition of Laplace matrices, aggregating the features of all neighbors of each node to update the new node representation. However, GCN's defects are obvious as all nodes need to participate in the convolution process, making it transductive and poorly scalable. GraphSAGE [17] is proposed to give a solution, aiming to train an aggregator rather than obtain a simple representation of each node. It updates node features by sampling a fixed number of neighbor nodes (not limited to first-order neighbors) and designing an aggregation function. This way, GraphSAGE can learn node features inductively. However, setting a fixed number of neighbor nodes may lead to poor node representation and low efficiency on graphs with high node density. GAT [18] utilizes the masked self-attention mechanism, which assigns different weights to different neighbor nodes when aggregating node features.

Existing studies for TUL have achieved promising results, but the problems discussed in Section 1 limit the effective application of their proposed models. In order to accomplish TUL tasks more effectively and efficiently, the novel model TULMGAT is developed by addressing these problems concurrently.

3. Preliminary

In this section, we first present the notations used throughout the paper in Table 2 and then introduce several definitions. Finally, we formulate the problem of this study.

Definition 1 (*Check-in Record*)**.** A check-in record $r = (u, l, t)$ refers to the geographic coordinate $l = (lng, lat)$ (i.e., longitude and latitude) generated by user u at timestamp t.

Definition 2 (*Grid Index*)**.** Given a grid granularity, the check-in space is divided into grid cells that can be indexed.

Definition 3 (*Trajectory*)**.** The sequence $\tau = (r_1, r_2, \ldots, r_n)$ generated by user u during a time interval is defined as a trajectory, where each r_i in τ is a check-in record.

Definition 4 (*Graph*)**.** Given a trajectory τ and pre-defined spatial rules, connect check-in records in the trajectory τ to obtain a graph G to represent τ.

In summary, after clarifying the limitations of previous TUL work, we construct a check-in-oriented graph called COG to explore features beyond the inherent sequential information of trajectories. GPS-based devices capture users' check-in records, and for spatial information within a time interval, we represent the check-in records as grid cells by Grid Index. By utilizing the inherent sequence relationships of the nodes within the trajectory and pre-defined spatial rules, we construct the COG to further explore the human mobility patterns. Details and examples of the construction of COG will be described in Section 4.1.2.

Problem Formulation. Given a set of anonymous trajectories $T = \{\tau_1, \tau_2, \ldots, \tau_n\}$ and the set of users $U = \{u_1, u_2, \ldots, u_m\}$ who generated them, the task of TUL is to learn a classifier to link these anonymous trajectories to their owners: $T \mapsto U$.

4. TULMGAT

In this section, we describe the architecture and technical details of TULMGAT. As shown in Fig. 3, it has four main modules: (1) **Construction of Check-in Oriented Graphs**: We construct multiple check-in oriented graph $COGs$ of different scales to mine node features based on the trajectory order and pre-defined graph knowledge. (2) **Node Embedding**: For each node of a COG, we embed it into a low-dimensional vector space with a graph attention network and apply global average pooling to aggregate the embeddings of an arbitrary number of nodes into a uniform dimension. (3) **Trajectory embedding**: A BiLSTM is designed to fuse information between scales (4) **Linking**: A multi-layer perceptron (MLP) and softmax function-based linking module is used to link anonymous trajectories to corresponding users.

4.1. Construction of check-in oriented graphs

4.1.1. Grid index and embedding

To study the spatio-temporal information of trajectories in more detail, we need to design a complete and generalized representation of trajectories first.

As we know from Section 3, a check-in record contains at least two-dimensional spatial information with a timestamp, and a trajectory is a collection of check-in records over a fixed time interval. To effectively extract the spatial features involved in the check-in records, we first process the check-in records as trajectories according to Definition 3, and then partition the geographic space by grid index according to Definition 2.

Grid index is designed to divide geographic information into grids for encoding and representation. For example, given a grid granularity, as shown in Fig. 4, the space is divided into $N \times N$ grid cells (N is assumed to be 4 here). The trajectory τ can then be denoted as a sequence of grid cells: $G(\tau) = (g_{11}, g_{12}, g_{12}, g_{23}, g_{13}, g_{24}, g_{33}, g_{34}, g_{43}, g_{31})$. Note that to preserve the real human mobility patterns in the context of the original trajectories, we do not sample and pad the check-in records in this process.

It should be noted that because the earth is an approximate ellipsoid, the size of each longitude and latitude is not the same. The distance of each degree of latitude is about 111 km, but the distance of each degree of longitude is $111cos\theta$ kilometres where θ is the latitude of the specific coordinates. Therefore, using a Grid Index with fixed grid granularity may cause unfair division on longitude. In navigation and other geographic location applications, complex cartographic rules and coding methods are often involved, but in Trajectory-User Linking such errors are acceptable.

Compared to the additional spatial location information or pre-processing required for POI clustering [10], Grid Index is a fast and effective way of spatial delineation. Moreover, the different densities

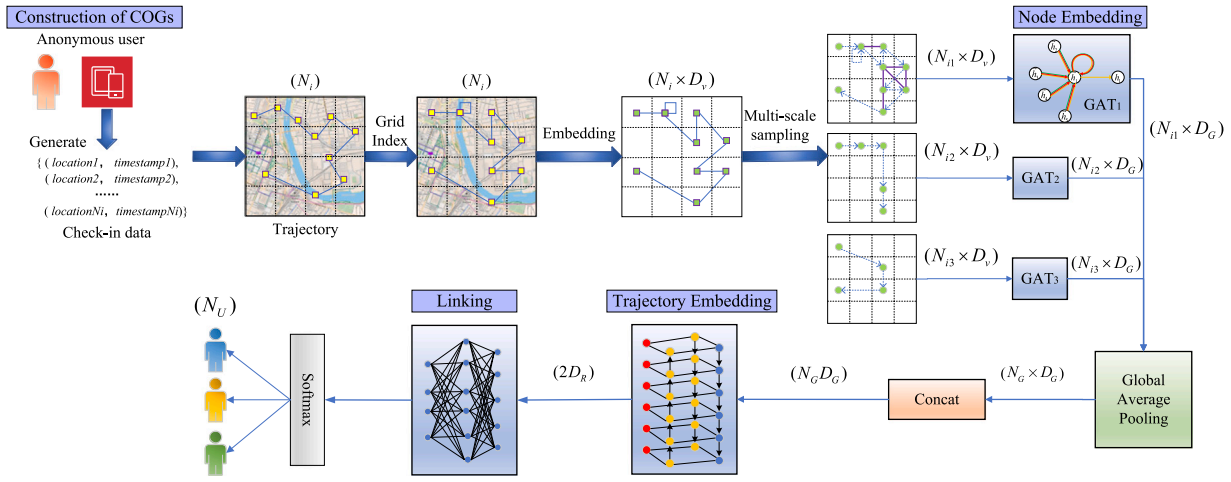

Fig. 3. The structure of TULMGAT.

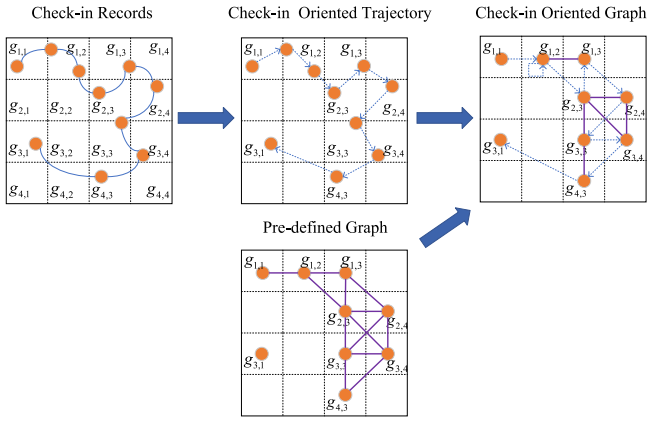

Fig. 4. An example of the construction of the graph *COG*.

of urban functional areas can cause sparsity in clustering POIs, leading to inconsistent and low sampling rates of trajectories. Therefore, we adopt a grid index to represent trajectories initially [26], which have been proven to be effective in representing spatial information in other work [16,27].

Embedding encodes grids as low-dimensional dense vectors, facilitating deep learning models to learn human mobility patterns in trajectories. The number of grids after check-in records or traffic datasets is very large. For instance, the Gowalla dataset used in our work has at least tens of thousands of effective grids after the grid index. Traditional bag-of-words models or one-hot encoding obviously cannot effectively represent the trajectory because of sparsity and other issues. Therefore, similar to TULER [4], we apply CBOW in Word2Vec [9] to process grids to obtain differentiated representation vectors. However, there is always a correlation between geographical locations because of the user's check-in behavior and spatial proximity. Compared to word embedding which treat grids as words and trajectories as sentences, randomly initialized node representations are not recommended and often fail to train due to illogical parameter settings.

4.1.2. Check-in oriented graph

Intuitively, trajectories are not only temporal sequences but also natural directed graphs. Recent work has shown that trajectories, especially those taken from geospatial space, should not be treated as pure temporal sequences, and there has been much success in exploring the valid information contained in trajectories from a non-Euclidean perspective and accomplishing downstream trajectory-related tasks [28].

However, the directed graphs directly translated from trajectories are sparse, with the number of edges equal to the number of nodes minus one ($|E| = |V| - 1$), which is obviously not conducive to making full use of the information about the node neighbors, for which we draw on pre-defined rules or metadata to extend the graph structure. In general, graph extension methods can be classified as:

- **Rule-based**: Euclidean distance or geodesic distance [29] can effectively represent the spatial relationship between nodes. If the distance between two nodes is considered to be less than a given threshold, we consider that there is a reliable edge relationship between nodes.
- **Metadata-based**: Metadata requires additional collection. Nodes can be considered to be connected if some additional attribute of two nodes is told to be the same or similar, such as road ID [16], user identification, and user behavioral interactions [30].

In our work, since there is no additional metadata, we choose a rule-based pre-defined graph construction method and apply Euclidean distances to determine the spatial relationships between nodes. This approach is similar to treating trajectories as images [7], but to address issues like image sparsity, we treat trajectories as non-Euclidean graphs to further learn trajectories features by graph neural networks.

$$e_{ij} = \begin{cases} 1 & if \ \ (x_i - x_m)^2 + (y_j - y_n)^2 \le \lambda, \\ 0 & else. \end{cases} \tag{1}$$

As shown in Eq (1), we take the Euclidean Distance to determine whether two grids $g_{i,j}$ and $g_{m,n}$ are connected, where x and y denote the grid subscripts and λ is the human-set threshold. In our work the threshold is set to 2, meaning a grid is connected to eight of its neighbors. Briefly speaking, if two grids $g_{i,j}$ and $g_{m,n}$ are in a trajectory, we connect them if they are adjacent in geographic space, even if they do not have a first-order neighbor relationship [31] that comes from the sequential nature of the trajectory. For instance, as shown in Fig. 4, g_{23} and g_{34} originally have no direct edge relationship, but we connect g_{23} and g_{34} with undirected edges based on rules.

Although the rule-based pre-defined graph may seem plain, it alleviates the shortcomings of the Grid Index. Data variation [32] refers to grids generated in the same area being considered as the same class but treated separately due to specific coordinate values and slight differences. Therefore, the rule-based pre-defined graph connecting adjacent grids actually addresses both sparsity and data variation in trajectory graph representation.

With this scheme, as demonstrated in Fig. 4, we process the Check-in Records collected at a time interval as grids. Then, the grids are directly processed into a sparse directed graph (Check-in Oriented Trajectory)

based on temporal order and a pre-defined graph based on Euclidean Distance. Finally, we merge the two graphs to obtain the Check-in Oriented Graph COG representing the original trajectory. A check-ins oriented graph COG is represented as $G(V, E, F)$: (i) V is the set of nodes in a trajectory, where grids represent the nodes; (ii) E is the set of edges between the grids; (iii) F is the set of initial embeddings of nodes by performing Word2vec on the grids in all trajectories.

4.1.3. Multi-scale sampling

Due to the natural randomness of user check-in behavior, the number of trajectories and nodes included in the trajectories are always different between users as shown in Fig. 2, which leads to models that tend to have unstable performance differences over trajectories of different lengths. As a result, we propose to process every single trajectory with different sampling rates to improve the robustness of our model.

With the overall graph structure maintained, we set multiple sampling rates, such as 100% (no sampling), 50% (sampling every two nodes), and 33% (sampling every three nodes) to delete nodes from the check-ins oriented trajectories and adjust the connections of the remaining nodes based on the pre-defined graph to obtain several different $COGs$ as illustrated in Fig. 3. However, it should be emphasized that the $COGs$ still collectively represent a single trajectory and simply enter the different GAT channels in Node Embedding.

4.2. Node embedding

Following the construction of the graphs $COGs$, we design a neural network module to learn the representation of geographic information in trajectories. From the way we construct $COGs$, the original sequential visits in the trajectories are designed as directed edges, and a rule-based pre-defined graph complements the spatial information and alleviates sparsity, which actually weakens the temporal properties of the human mobility pattern. However, recent research work [27,33] has demonstrated that the spatial properties of trajectories actually play a more critical role. Therefore, optimizing the node representation in trajectories by graph neural network is crucial in TULMGAT.

Graph attention network (GAT) module optimizes node representation in $COGs$. Each GAT processes the COG at the corresponding sampling rate, with several GATs in parallel throughout Node Embedding.

Assume the input of each graph attention layer is a set of node vectors $\mathbf{h} = \{\mathbf{h}_1, \mathbf{h}_2, \dots, \mathbf{h}_{N_i}\}$, $\mathbf{h}_i \in R^{D_v}$, where N_i represents the arbitrary number of nodes in a COG and D_v is the initial embedding dimension of the node. In our work, the set of embeddings of nodes F obtained by Word2vec is utilized as the initial input of GAT. The output of each graph attention layer is a new set of node vectors $\mathbf{h}' = \{\mathbf{h}_1', \mathbf{h}_2', \dots, \mathbf{h}_N'\}, \mathbf{h}_i' \in R^{D_G}$, where D_G is the dimension of the new vector, and the details of obtaining $\mathbf{h}'$ are discussed as follows.

Although GNNTUL [15] based on GraphSAGE [17] exploits node features, it suffers from the following problem: (i). When aggregating node features, it does not consider the different importance of neighboring nodes, ignoring interaction information between different check-in records from a spatial perspective. (ii). GraphSAGE samples a fixed number of neighboring nodes, which results in serious information loss for trajectories with an uncertain number of nodes. Different from GNNTUL, our proposed model, TULMGAT, effectively solves the problems by applying the graph attention network with masked multi-head self-attention to learn nodes representations. Similar to the general attention mechanism, GAT calculates the attention value e_{ij} between source node v_i and neighbor node v_j based on their representation vectors:

$$e_{ij} = LeakyReLU(\mathbf{a}^T(W\mathbf{h}_i \parallel W\mathbf{h}_j)) \tag{2}$$

where $\mathbf{a}$ is a $R^{D_G \times D_G}$ mapping which is implemented by a single-layer feed-forward neural network, $W \in R^{D_v \times D_G}$ is a weight matrix shared by all $\mathbf{h}_i$. Moreover, we apply $LeakyReLU$ to perform nonlinear activation for each pair of concatenated $W\mathbf{h}$ to ensure that the directed attention relation has been learned and prevent the node's own features $\mathbf{h}_i$ from being smoothed out by the subsequent normalization.

In general, transductive graph neural networks such as GCN require updating all known nodes simultaneously, and the self-attention mechanism tends to assign attention to all known elements. However, since each trajectory contains only a few nodes, observing all nodes of all trajectories to update a single node within a single trajectory has high time complexity. Furthermore, it can lead to the loss of unique structural information within a single trajectory due to over-computation. Therefore, we adopt the masked attention mechanism, where each node computes attention to only its own neighbor nodes within a single trajectory:

$$\begin{aligned} a_{ij} &= softmax(e_{ij}) \\ &= \frac{exp(LeakyReLU(\mathbf{a}^T[W\mathbf{h}_i \parallel W\mathbf{h}_j]))}{\sum_{k \in \mathcal{N}_i} exp(LeakyReLU(\mathbf{a}^T[W\mathbf{h}_i \parallel W\mathbf{h}_k]))} \\ &= \frac{exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} exp(e_{ik})} \end{aligned} \tag{3}$$

where $\mathcal{N}_i$ is the set of neighbor nodes of v_i in a single trajectory. To obtain accurate and stable inter-node relationships in self-attention calculation, we introduce a multi-head self-attention mechanism to improve the model's ability to characterize the spatial relationships between nodes. For the output of the middle layer, we apply K different W to calculate the attention and derive the output vector $\mathbf{h}_i'$ by concatenating:

$$\mathbf{h}_i' = \parallel_{k=1}^{K} \sigma\Big(\sum_{j \in \mathcal{N}_i} a_{ij}^k W^k \mathbf{h}_j\Big) \tag{4}$$

Notably, the number of $\mathbf{h}_i'$ in $\mathbf{h}'$ computed for different trajectories is variable due to different amount of geographic locations contained in trajectories. To ensure all nodes in a trajectory are considered and form a uniform dimensional vector for the next network layer, we apply Global Average Pooling, which replaces the final fully connected layer in the graph attention network. Global average pooling averages the feature vectors of all nodes and aggregates them to a fixed dimension. As there is no learnable parameter, the global average pooling can significantly alleviate potential overfitting problem. In the final stage of GAT, we feed $\mathbf{h}'$ into the global average pooling to get the trajectory representation $\mathbf{e}_\tau$ at a single scale:

$$\mathbf{e}_\tau = Average(\mathbf{h}_1', \mathbf{h}_2', \dots, \mathbf{h}_N'), \mathbf{e}_\tau \in R^{D_G} \tag{5}$$

Following this way, we generate several $COGs$ with each sampling scale for the same trajectory and concatenate the trajectory representations $\mathbf{e}_\tau^k$ of N_G COGs as the final trajectory representation of this module.

$$\mathbf{e}_\tau = \parallel_{k=1}^{N_G} \mathbf{e}_\tau^k \tag{6}$$

4.3. Trajectory embedding

After obtaining the representation of multiple $COGs$ for a single trajectory, we flatten $\mathbf{e}_\tau \in R^{N_G D_G}$ to represent the trajectory and apply an RNN-based model to mine the interaction of different scales.

Distinguish from previous TUL methods in which RNN-based model is designed for mining the sequential relationships between node embeddings within a trajectory, we apply the bidirectional LSTM model to capture the relationships between representations of the same trajectory at different scales due to sampling rates, which has no explicit directional temporal dependence.

As an extension of RNN, LSTM introduces memory cells with different structures. The LSTM cell at each moment contains an input gate i_t, a forget gate f_t, and an output gate o_t. The input of the cell at current

moment will include the current input x_t, the hidden state h_{t-1} and the cell state c_{t-1} of the previous moment:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{7}$$

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{8}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{9}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot tahn(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{10}$$

$$h_t = o_t \odot tahn(c_t) \tag{11}$$

where all $W \in R^{N_G D_G \times D_R}$ and b are learnable parameters, $\odot$ is the symbol for Hadamard product, σ, $tahn$ are activation functions, and the initial input x for BiLSTM is $\mathbf{e}_\tau$. At time t, the output of BiLSTM to the fully connected neural network is to concatenate the hidden state h_t of the forward LSTM and the hidden state $\hat{h}_t$ of the backward LSTM as the ultimate trajectory representation:

$$\mathbf{e}'_\tau = Concat(h_t, \hat{h}_t) \tag{12}$$

It should be made clear that trajectory embedding is never the focus of our work, but replacing it with a simple projection in ablation experiments in Section 5.3.4 still degrades performance a bit. Moreover, time complexity experiments in Section 5.2.3 will show that the main training time of TULMGAT is spent on this and later linking layers, and that more advanced but complex coding methods such as dilated temporal convolution, attention network, or even Transformer are certainly worth considering because of accuracy, but can be detrimental to the computational complexity we are aiming for, and the RNN-based models have always maintained the right balance between accuracy and training time for Trajectory User Linking.

4.4. Trajectory-user linking

Generally, TUL can be regarded as a classification problem in which trajectories are classified into individual users. A Multi-Layer Perceptron (MLP) and Softmax function based network is proposed to link trajectories to the users who generated them. The ultimate representation $\mathbf{e}'_\tau$ of trajectory τ is fed into MLP:

$$Y = W\mathbf{e}'_\tau + b \tag{13}$$

where $W \in R^{D_R \times N_U}$ and b are the learnable parameters in the multi-layer perceptron. Assume the output of MLP $Y = \{y_1, y_2, \ldots, y_n\}$, we calculate the probability that τ is generated by u_i through Softmax function:

$$p(y_i) = \frac{exp(y_i)}{\sum_{j=1}^{n} exp(y_j)} \tag{14}$$

4.5. Training settings

In the node embedding module, we apply a two-layer graph attention network to learn node representations. Moreover, we use $LeakyReLU$ activation in the update process of multi-headed self-attention layer and $ReLU$ activation between graph attention layers. In addition, we set up dropout layers to mitigate the over-fitting problem. In the trajectory embedding module, we initialize the hidden states and cell states of the BiLSTM with all-zero initialization. In the linking module, we employ the *cross-entropy* function as the loss function and adopt $Adam$ as the optimizer.

5. Experiment

In this section, we verify the performance of the proposed model TULMGAT in two aspects, including comparative experiments and validity experiments. In comparative experiment, we present the statistics of datasets, evaluation metrics, baselines, and conduct extensive experiments with existing mainstream models on publicly available datasets. In model validity experiments, we will test the robustness of TULMGAT's variants, compare GAT with other graph neural networks, examine the performances under single-scale trajectories, and validate the effectiveness of each module through ablation experiments. In addition, the effect of main parameters and the effectiveness of the multi-scale trajectory sampling scheme will be examined by comparing it with TULGAT, which does not adopt multi-scale trajectory sampling. Our code is accessible.[1]

Table 3
Statistics of datasets.

Datasets	$\|U\|$	$\|T\|$	$\|G\|$	$\|Avg\|$	$\|E\|$	$\|C\|$
Gowalla	247	14,780	20,132	11.85	115,364	175,177
	149	8976	10,433	11.61	61,679	104,240
Brightkite	223	23,938	13,601	8.60	84,315	205,859
	141	15,739	9273	8.34	55,738	131,243
Foursquare-NYC	1083	49,769	17,652	1.93	150 943	95,887
Foursquare-TKY	2293	118,252	20,597	2.45	252 197	289,978

5.1. Datasets and evaluation metrics

We perform all experiments on two publicly available datasets: Gowalla[2] and Brightkite.[3] Moreover, we add the public datasets Foursquare[4] in comparative experiment to enhance the persuasiveness.

As shown in Section 4.1.1, we represent check-in records as grids according to Grid Index and embed the grids of the training dataset as vectors by CBOW. In this way, we represent sequential trajectories of varying lengths according to the daily check-in records, while whether the trajectories are sampled with a fixed length depends on the specific model. Furthermore, based on the methods in Sections 4.1.2 and 4.1.3, a single trajectory is further processed into check-in oriented graphs $COGs$ for GNN-based methods.

The statistics of datasets are summarized in Table 3, where $|U|$ denotes the number of users, $|T|$ is the number of trajectories, $|G|$ is the number of grid cells that contain at least one check-in record, $|Avg|$ is the average number of check-in records per trajectory, $|E|$ is the number of all unique edges in all graphs $COGs$. $|C|$ is the number of all check-in records involved. It should be noted that due to open source issues and different methods of pre-processing location information, we are unable to standardize the dataset settings with previous work. For Foursquare, we apply two open source datasets in New York(NYC) and Tokyo(TKY). For all datasets, we sequentially intercept some of the data for experimentation and use one day as the time interval for the trajectories. Referring to the parameter settings in [27] and [16], we select the appropriate grid granularity for each dataset to obtain the above data.

Similar to previous work [4,14], we employ ACC@1, ACC@5, macro-P, macro-R and macro-F1 as evaluation metrics. ACC@K indicates the classification accuracy of user linking, and Macro-F1 is the average F1 value of all user categories or the harmonic mean of precision (macro-P) and recall (macro-R):

$$ACC@K = \frac{\#correctlyidentifiedtrajectories@K}{\#trajectories} \tag{15}$$

$$Macro\text{-}F1 = \frac{2 \times macro\text{-}P \times macro\text{-}R}{macro\text{-}P + macro\text{-}P} \tag{16}$$

[1] https://github.com/blisky-li/TULMGAT.
[2] https://snap.stanford.edu/data/loc-Gowalla.html.
[3] https://snap.stanford.edu/data/loc-brightkite.html.
[4] https://sites.google.com/site/yangdingqi/home/foursquare-dataset.

Table 4
Parameters details.

Parameter	Available range	Recommended value
Vector size(D_v)	[128,300]	128
Hidden size in baselines	[300, 1000]	500
Learning rate	[0.0025, 0.00095]	0.0025
Batch size	[32, 128]	64
Dropout rate	[0, 0.55]	0.25
Stacked GAT	[2, 4]	2
D_G	[16, 64]	32
D_R	[32, 128]	64
Number of heads	[6,12]	12

5.2. Comparative experiments

5.2.1. Baselines and parameter setting

We compare our model TULMGAT with the following TUL baselines:

- **TULER** [4]. TULER is the first model to solve the task TUL, and utilizes various RNNs to learn user mobility and implement classification tasks. We applied variants of TULERs for the comparative experiments, including TULER-LSTM, TULER-GRU, and BiTULER.
- **TULVAE** [11]. The model learns human mobility patterns in a semi-supervised manner, and attempts to mine deep semantic information in trajectories through Variational Autoencoder.
- **TULAR** [14]. TULAR is the first work to focus on information loss due to sampling. It proposes an unsupervised TSV module to aggregate trajectories of arbitrary length after an RNN-based model and applies an attention mechanism to filter trajectory distribution information.
- **STULIG** [12]. STULIG adds hierarchical potential factors to TULVAE to examine the characteristics of trajectory distribution from multiple perspectives, and extends the training data by generating synthetic yet plausible trajectories.
- **GNNTUL** [15]. GNNTUL is the first model that treats trajectories as graph and applies graph neural networks. It samples a fixed number of nodes based on the GraphSAGE model and updates node embeddings based on neighboring nodes for user unique mobility mining.
- **TULRN** [16]. TULRN is an improvement of GNNTUL, it replaces the location information in GNNTUL with a priori road network and introduces k-Renyi entropy to calculate the weights of nodes of trajectories in training dataset. In this way, weighted neighbor sampling is performed on nodes in each unknown trajectory instead of random sampling in GraphSAGE.

For our model TULMGAT, we set three sampling rate strategy(A: 100%/ No sampling; B: 50%/ Sample every two nodes; C: 33%/ Sample every three nodes) and obtain four variants about TULMAT: (1) TULMGAT-O: Strategy A. Due to no sampling of the original trajectory, TULMGAT-O is also called TULGAT. (2) TULMGAT-T: Strategy A+B. (3) TULMGAT-TH: Strategy A+B+C. They concatenates the output of global average pooling. (4) TULMGAT-Add: The output of the global average pooling is directly accumulated as the trajectory representation on the basis of TULMGAT-TH.

All experiments are performed on trajectory data divided based on grid index, and the embedding size of trajectory nodes of Word2vec is 128. Other experimental parameters for baselines follow the settings reported in their papers and we emphasize that the sample length of their trajectories is always 9. For TULMGAT, the optimizer is Adam with a learning rate of 0.0025, the embedding dimension of single head of GAT is set to 32, and the dropout rate is 0.25. More sufficient details of parameters are given in Table 4.

5.2.2. Experimental analysis

The experimental results of all approaches on given datasets are presented in Table 5, the best experimental results are marked boldly, and the best results in baselines are underlined. To be clear, we could not record the TULVAE and STULIG results on Foursquare datasets because they are just too time consuming.

Observed from this, the proposed model TULMGAT performs better than baselines in all metrics. On Gowalla |149|, compared to GNNTUL which is the best model in baselines in terms of ACC@1, TULMGAT-T achieves the best results yielding 6.08%, 2.72%, 5.33%, 5.12%, and 5.23% improvements for ACC@1, ACC@5, macro-P, macro-R and macro-F1 metrics. On Gowalla |247|, TULMGAT-TH achieves the best results in terms of ACC@1, yielding 4.44%, 1.30%, 2.66%, 3.27%, and 2.99% improvements compared to TULRN, which is the best model in baselines. On Brightkite |141|, TULMGAT-TH achieves the best results in terms of ACC@1, yielding 4.88%, 1.22%, 5.28%, 7.09%, and 6.22% improvements compared to TULRN. On Brightkite |223|, TULMGAT-O achieves the best results in terms of ACC@1, yielding 4.36%, 1.67%, 5.98%, 6.59%, and 6.30% improvements compared to TULRN for ACC@1, ACC@5, macro-P, macro-R and macro-F1 metrics. Undoubtedly, our work on Foursquare also achieves the best results. Our best model TULMGAT-TH on Foursquare-NYC outperforms TULRN on five metrics by 7.39%, 6.34%, 6.01%, 7.91% and 7.02%, and TULMGAT-TH on Foursquare-TKY outperforms GNNTUL by 5.65%, 7.26%, 6.64%, 7.19% and 6.94% on the five metrics.

Among the variants of TULMGAT, it is clear that the performance of TULMGAT-Add is significantly weaker than the other variants because the information on different scales is accumulated after global average pooling, which undoubtedly leads to information coupling. Therefore, TULMGAT-T and TULMGAT-TH can usually outperform TULMGAT-O (TULGAT), indicating that mining trajectory information from different sampling scales is effective, but there are still subtle differences because of different datasets. Overall, the model performance on multi-scale trajectory sampling is excellent and stable, meaning that it can converge faster to a result we can accept. Specifically, TULMGAT performs better for those reasons:

(1) TULMGAT effectively represents trajectories as graph structures and introduces graph neural network learning, which fully explores the features of geographic locations and their interactions;

(2) TULMGAT employs global average pooling to achieve the application of all geographic locations in trajectories, which fully exploits the complete context information of trajectories. **Significantly**, the trajectories in Gowalla and Brightkite are long as shown in Table 3 and larger than the number of samples in baselines, but trajectories in Foursquare tend to be shorter, so we attribute the failure of baselines to the fact that the former is missing trajectory information while the latter fills in too much redundancy resulting in feature smoothing;

(3) TULMGAT utilizes a multi-head self-attention mechanism to effectively mine the deep semantic information and user mobility patterns of trajectories.

(4) TULMGAT samples trajectories at multiple scales and uses an RNN-based model to mine the interactions between the scales, which learns the representation of trajectories from different spatial perspectives while preserving all the original information.

5.2.3. Computational complexity analysis

Next, we analyze the computational complexity of each model. For descriptive convenience, we use n to denote the number of original or sampled nodes in the trajectory, which do not differ significantly. E denotes the original embedding of a node and F represents the embedding of a node after feature extractor. In addition, e refers to the number of edges in the trajectory graph. We should notice that $n < F < E$ and $e \ll n^2$ in most instances.

Computational complexity of each model after omitting MLP classifier are shown in Table 6. The computational complexity of the models represented by TULER comes mainly from the RNN model.

Table 5
Comparison of TULMGAT with others methods on different datasets.

Gowalla										
Method	$\|U\| = 149$					$\|U\| = 247$				
	ACC@1	ACC@5	macro-P	macro-R	macro-F1	ACC@1	ACC@5	macro-P	macro-R	macro-F1
TULER-LSTM	45.52%	64.32%	42.93%	35.71%	38.99%	42.87%	60.12%	40.08%	32.64%	35.98%
TULER-GRU	44.98%	64.18%	41.97%	35.58%	38.51%	42.09%	59.74%	38.83%	32.92%	35.63%
BiTULER	45.86%	64.47%	43.07%	35.88%	39.15%	43.36%	60.34%	39.98%	33.71%	36.58%
TULVAE	49.57%	69.72%	46.28%	41.68%	43.86%	47.82%	64.03%	42.88%	39.04%	40.87%
TULAR	54.24%	71.89%	51.17%	46.80%	48.89%	52.84%	69.42%	50.19%	47.84%	48.99%
STULIG	51.98%	70.64%	48.46%	42.93%	45.53%	49.69%	66.74%	44.85%	41.76%	43.25%
GNNTUL	54.64%	71.16%	51.31%	47.44%	49.30%	52.65%	68.82%	49.51%	47.06%	48.25%
TULRN	54.46%	70.76%	51.07%	48.13%	49.56%	54.04%	69.81%	50.66%	47.52%	49.04%
TULMGAT-O	60.09%	73.32%	53.95%	50.65%	52.25%	57.80%	70.73%	53.58%	51.05%	52.28%
TULMGAT-T	**60.72%**	**73.88%**	**56.64%**	**52.56%**	**54.53%**	58.39%	**71.47%**	54.15%	**51.43%**	**52.76%**
TULMGAT-TH	58.39%	71.47%	54.15%	51.43%	52.76%	**58.48%**	71.11%	53.32%	50.79%	52.03%
TULMGAT-Add	59.57%	72.50%	54.05%	50.91%	52.43%	57.44%	71.22%	**54.38%**	49.17%	51.64%
Brightkite										
Method	$\|U\| = 141$					$\|U\| = 223$				
	ACC@1	ACC@5	macro-P	macro-R	macro-F1	ACC@1	ACC@5	macro-P	macro-R	macro-F1
TULER-LSTM	55.05%	67.35%	50.66%	44.62%	47.45%	54.21%	66.42%	49.84%	43.79%	46.62%
TULER-GRU	55.45%	67.82%	51.89%	44.29%	47.79%	54.89%	66.06%	50.51%	44.88%	47.53%
BiTULER	55.62%	67.93%	51.81%	44.78%	48.04%	54.87%	67.06%	50.36%	44.29%	47.13%
TULVAE	61.87%	74.83%	57.41%	52.09%	54.62%	60.23%	72.12%	55.28%	51.75%	53.46%
TULAR	69.83%	83.47%	63.19%	58.45%	60.73%	68.78%	82.19%	61.67%	58.40%	59.99%
STULIG	65.98%	78.56%	60.69%	55.87%	58.18%	65.30%	77.64%	59.43%	54.92%	57.09%
GNNTUL	70.26%	83.61%	62.63%	59.80%	61.18%	69.39%	82.58%	62.23%	59.01%	60.58%
TULRN	70.93%	84.10%	64.28%	60.34%	62.25%	70.90%	83.15%	63.40%	59.67%	61.48%
TULMGAT-O	75.34%	85.28%	**70.33%**	66.55%	68.39%	**75.26%**	**84.82%**	69.38%	66.26%	67.78%
TULMGAT-T	75.66%	85.28%	70.26%	66.80%	**68.48%**	74.78%	84.61%	**69.92%**	**66.39%**	**68.11%**
TULMGAT-TH	**75.81%**	**85.32%**	69.56%	**67.43%**	68.47%	74.85%	84.35%	68.35%	66.28%	67.30%
TULMGAT-Add	75.07%	85.09%	70.24%	66.43%	68.28%	74.44%	84.53%	69.10%	65.83%	67.43%
Foursquare										
Method	NYC					TKY				
	ACC@1	ACC@5	macro-P	macro-R	macro-F1	ACC@1	ACC@5	macro-P	macro-R	macro-F1
TULER-LSTM	45.85%	58.00%	41.14%	36.10%	38.46%	36.06%	47.85%	25.92%	22.09%	23.85%
TULER-GRU	43.87%	57.98%	40.88%	35.96%	38.26%	35.32%	48.21%	25.48%	22.29%	23.78%
BiTULER	46.71%	58.62%	42.04%	36.52%	39.09%	37.28%	49.17%	27.91%	24.72%	26.22%
TULVAE	–	–	–	–	–	–	–	–	–	–
TULAR	51.49%	63.28%	47.89%	44.91%	46.35%	40.89%	52.99%	30.68%	29.02%	29.83%
STULIG	–	–	–	–	–	–	–	–	–	–
GNNTUL	52.88%	64.83%	46.87%	44.32%	45.56%	41.79%	53.61%	31.90%	29.73%	30.77%
TULRN	53.01%	65.03%	47.15%	43.98%	45.50%	41.59%	53.56%	31.36%	28.69%	29.97%
TULMGAT-O	60.22%	71.40%	52.47%	51.57%	52.01%	46.91%	60.82%	38.30%	36.50%	37.38%
TULMGAT-T	60.33%	**71.94%**	52.75%	51.56%	52.15%	47.10%	**60.94%**	38.25%	36.50%	37.35%
TULMGAT-TH	**60.40%**	71.37%	**53.16%**	**51.89%**	**52.52%**	**47.44%**	60.87%	**38.54%**	**36.92%**	**37.71%**
TULMGAT-Add	59.99%	71.25%	52.71%	51.49%	52.10%	46.83%	60.72%	38.22%	36.28%	37.22%

Table 6
Computational complexity of each model.

Model	Computational complexity
TULER-LSTM & TULER-GRU & BiTULER	$O(n^2EF)$
TULVAE & STULIG	$O(n^2EF + n^2EF)$
TULAR	$O(n^2E + nEF)$
GNNTUL & TULRN	$O(n^2EF + nF^2)$
TULMGAT	$O(nEF + eF + F^2)$

TULVAE and STULIG have additional computational overhead from the decoder because of the application of the generator. TULAR requires the computation of attention, but TSV reduces the dimensionality of subsequent vectors by averaging the node information. The computational cost of GNNTUL and TULRN are derived from RNN and aggregation operation in GraphSAGE. The GAT part of TULMGAT has strong parallel computation capability and global average pooling reduces the vector dimension of subsequent trajectory embeddings. The theoretical calculation reveals that our work possesses the lowest computational complexity and we are able to parallelize the attention computation for n nodes in the $O(nEF)$ part which is not possible for the RNN-based model.

The training hours for all models on NVIDIA TESLA PCIe V100S 32G are presented in Table 7, where we report results on Gowalla |247|, Brightkite |223| and two Foursquare datasets. Generative models like TULVAE and STULIG are very time-consuming because they require the construction of latent distributions to mine the semantics of trajectories. Moreover, the loss functions generated by their generators and classifiers are difficult to converge quickly, and we could not run them because it was too time consuming on Foursquare datasets. GNNTUL has the lowest runtime among all baselines because it samples a fixed number of neighbors to update node features, and the learning of graph structures tends to be highly parallel. The training time of TULRN is slightly longer compared to GNNTUL because of the extra edge weight calculation and screening. Unsurprisingly, our proposed model, TULMGAT, is far more efficient than all the baselines. This is because the computation of multi-head attention is independent and parallel, and the global average pooling can compress an arbitrary number of node embeddings into a low-dimensional trajectory representation, which greatly reduces the time complexity of learning user mobility from trajectories.

Moreover, in the TULMGAT variants, there is a subtle difference in training time for models that use graph structures at different scales.

Table 7
Training hours of all methods on NVIDIA V100 32G (h).

Dataset	BiTULER	TULVAE	STULIG	TULAR	GNNTUL	TULRN	TULMGAT-TH	TULMGAT-T	TULMGAT-O
Gowalla |247|	1.25	12.65	9.5	1.75	0.67	0.85	**0.20**	**0.19**	**0.17**
Brightkite |223|	1.45	13.9	11.05	2.25	1.05	1.21	**0.34**	**0.32**	**0.28**
Foursquare NYC	4.07	–	–	4.82	3.25	3.75	**0.90**	**0.82**	**0.75**
Foursquare TKY	8.30	–	–	10.32	7.25	8.50	**2.32**	**2.17**	**2.03**

TULMGAT-O, which uses only the original graph structure, shows more than 15% improvement in training time compared to TULMGAT-TH, which utilizes graphs at all three scales. In reality, whether to sacrifice some time to improve the robustness of the model as well as a small amount of accuracy depends on the data quality and data volume in the specific application scenario. However, the differences in training time among TULMGAT variants come from the number of GAT networks, and the gaps are small, suggesting that the main training time is spent on the Trajectory Embedding and Linking modules.

In conclusion, the core starting point of our work lies in a clearer and more detailed understanding of the spatio-temporal information of trajectories. The main problems of the previous work in terms of technical details are missing context of original trajectories, ignorance of spatial information and high computational complexity. Focusing on these three problems involving trajectory perception, we can achieve better results compared to previous work:

- The advantage of TULMGAT over GNNTUL and TULRN which also treat a trajectory as a graph and apply graph neural networks, lies in being able to process context information for trajectories of any length, and the performance of both shows that retaining all context in the original trajectory rather than sampling or filling it is a correct perception of trajectories.
- The advantage of our work over TULAR is the effective mining of the spatial information of trajectories, which proves that ignoring the spatial information of trajectories is a lack of consideration, and the effective use of the spatial information of trajectories is the key to linking trajectory users.
- Compared too previous works, we achieve lower training time by focusing on the spatial characteristics in trajectories. Subsequently, we can compute the information of the trajectories effectively and concisely in the node embedding phase by embedding the node information into a more low-dimensional space and by using parallel computation in the multi-headed attention computation, which makes the end-to-end deployment and application of trajectory user links more quickly and easily.

5.3. Model validity experiment

5.3.1. Robustness study

In robustness study, we will test the performance of three TULMGAT variants on trajectories of different lengths. As we state in Section 1, the main difference between Trajectory-User Linking and general classification tasks is that not only the number of trajectories per user is different, but also the length of each trajectory, i.e. the amount of information in single trajectory is different as well, which is similar to image distortion in image classification tasks. Therefore testing the robustness of the model against such noise that cannot be predicted in advance is required to apply the technique in reality.

We test the performance of variants of TULMGAT for ACC@1 on trajectories of different lengths and record the performance for which the number of trajectories is greater than 10. Moreover, we record the standard deviation of each model to facilitate performance comparisons.

As shown in Fig. 5, although the variants have essentially the same performance in Table 5, their accuracy varies significantly with trajectories of different lengths. TULMGAT-O has a obvious performance degradation compared to the other two models when facing very short trajectories, especially on Gowalla. Moreover, it may have serious classification failures on all datasets when facing certain lengths. Comparing the standard deviations, TULMGAT-TH consistently has the most stable performance, while TULMGAT-O tends to have poorer robustness. Although its standard deviation is slightly lower thanTULMGAT-T on Brightkite, this is because TULMGAT-T suddenly achieves better classification accuracy for some trajectory lengths. Therefore, we conclude that while multi-scale graph construction does not always guarantee the best classification performance for Trajectory-User Linking, TULMGAT can maintain better robustness across different trajectory lengths by fusing information from multi-scale graphs, and its classification confidence is often higher than the original TULGAT (TULMGAT-O).

5.3.2. GNN study

In this section, we investigate different graph neural networks. TULMGAT is based on the graph attention network(GAT) [18]. However, other common graph neural networks include graph convolutional networks(GCN) [25] and graph sample and aggregate (GrpahSAGE) [17]. GCN optimizes the information of each source node based on learning all neighboring nodes through a mapping function, and GrpahSAGE updates the information of source nodes by sampling a fixed number of nodes and learning an aggregation function. However, The GAT module we apply computes self-attention scores among nodes to learn how to autonomously perform aggregation of information from biased nodes.

In our GNN study, we replace the GAT module with GCN or GraphSAGE based on TULMGAT-TH and name them TUL-MGCN and TUL-MSAGE. Furthermore, because the GAT in TULMGAT contains a two-layer structure, we set up one or two layers of GCN and GprahSAGE for models. Their main parameters are consistent with TULMGAT.

As shown in Fig. 6, TULMGAT outperforms the other models in all metrics across all datasets, indicating that GAT is more capable of learning the effective user mobility compared to other graph neural networks. We attribute this to the fact that trajectories as sparse subgraphs in which the association between geographic locations is extremely unique, and the self-attentive mechanism can effectively learn this unique association. In contrast, GCN and GraphSAGE, especially their one-layer versions, struggle to extract this uniqueness.

5.3.3. Single-scale experiment

In this section we analyze the single sampling scale, which means that the whole network will only be experimented on one single COG transformed from one trajectory. To ensure the reasonableness of the trajectory representation, three different sampling rates are chosen: 100% (no sampling) 50% (sample every two nodes) 33% (sample every three nodes), and the corresponding models are called TULGAT-O, TULGAT-T and TULGAT-TH.

As shown in Fig. 7, it is clear that TULGAT-O outperforms the other models in terms of each metric, demonstrating that it is more effective to represent the trajectory and learn spatio-temporal information in the trajectory without sampling, which indicates the limitations of the models that require a fixed number of nodes to be sampled in the previous TUL work.

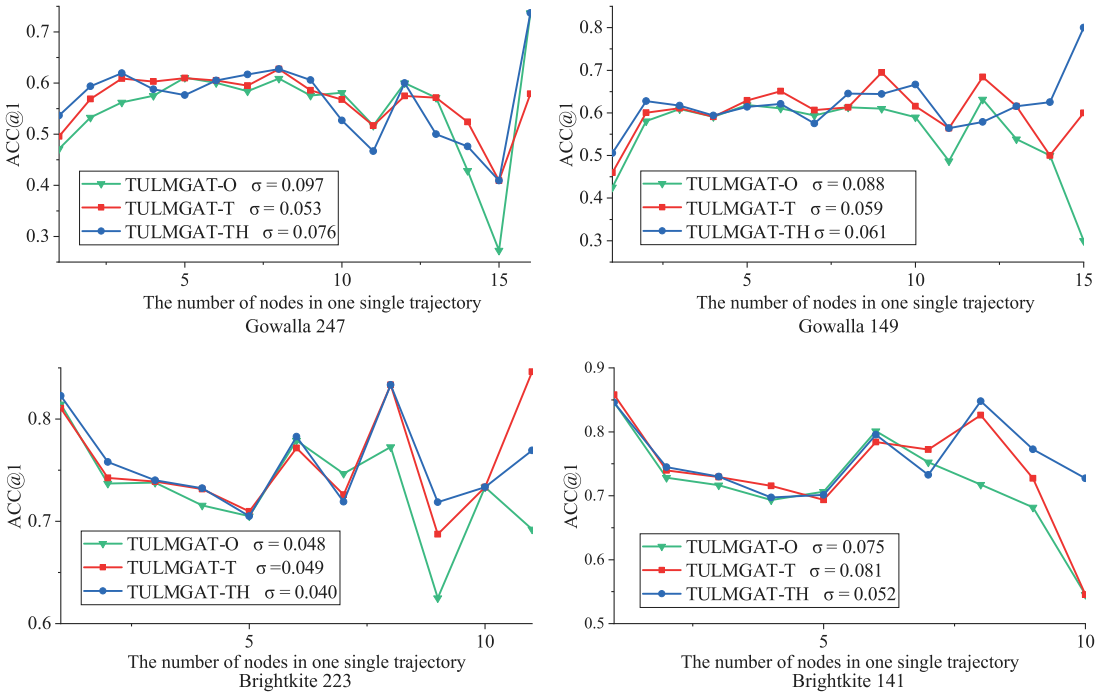

Fig. 5. Performance of TULMGAT on trajectories with different number of nodes.

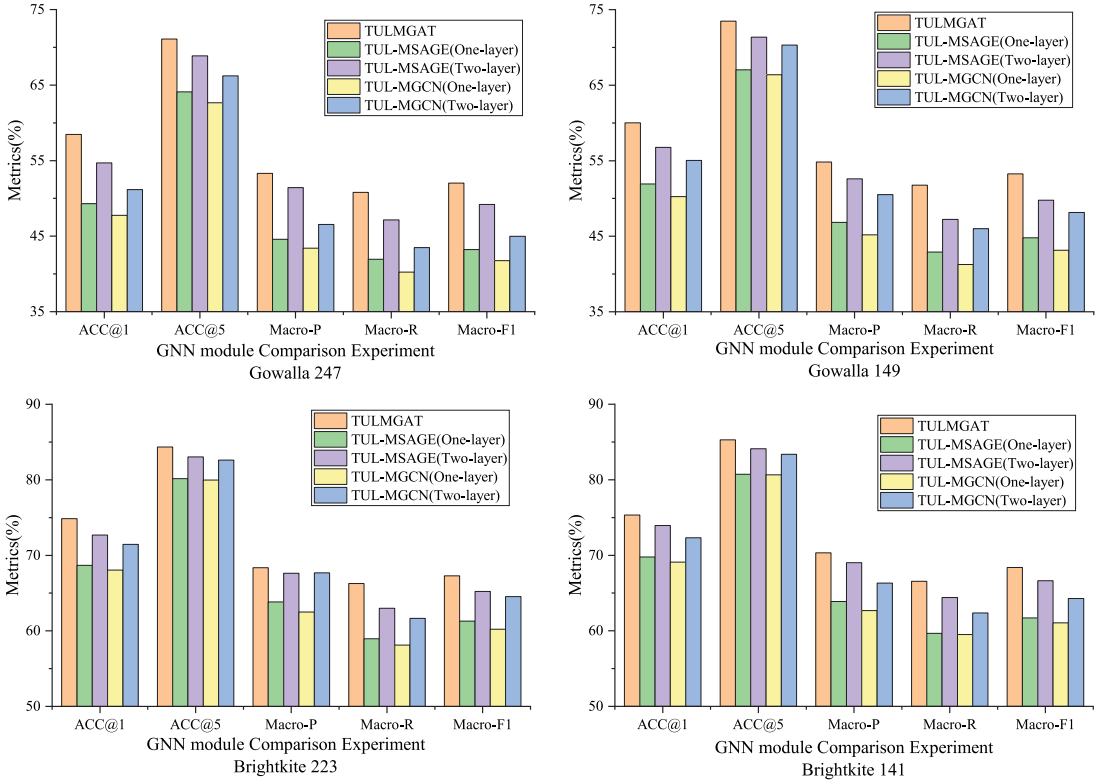

Fig. 6. GNN study.

5.3.4. Ablation study

To evaluate the impact of each main component of TULMGAT, we conducted experiments on several modules of TULMGAT. On the basis of TULMGAT-TH, we design different variants for ablation study:

- TULMGAT w/o MH: We set the multi-headed attention in the graph attention network to single-headed attention.
- TULMGAT w/o GAT: We eliminate the graph attention network module, and the initial node representations in the *COGs* will be fed directly to the global average pooling.
- TULMGAT w/o MS: Instead of sampling, we will feed the *COG* generated from a single complete trajectory alone into the model, which is also called TULGAT.

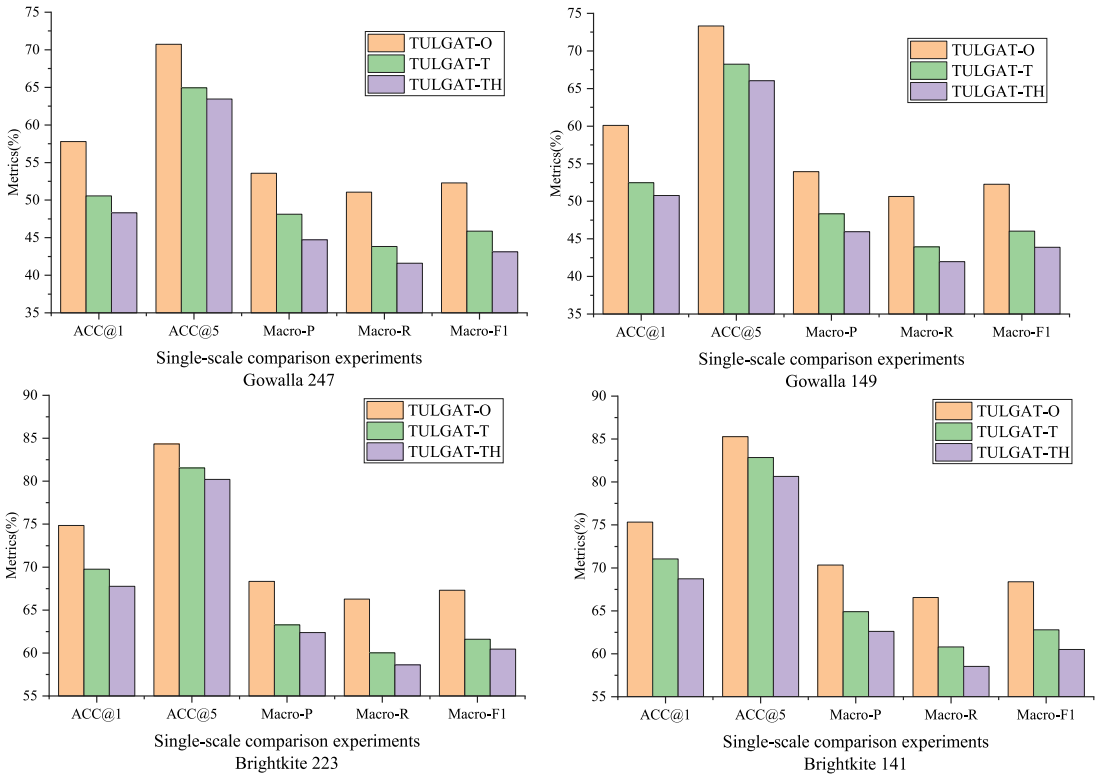

Fig. 7. Single-scale study.

- TULMGAT w/o BiLSTM: In trajectory embedding, we replace BiLSTM with directional LSTM.
- TULMGAT w/o LSTM: We replace the BiLSTM with a simple fully connected projection.
- TULMGAT w/o MS & LSTM: We discard both multi-scale trajectory sampling and RNN models.
- TULMGAT w/o PG: We remove the edge relations of rule-based pre-defined graph in Section 4.1.2 and Fig. 4, meaning the only edge relations in trajectories are those provided by the order of check-ins.

According to Fig. 8, TULMGAT w/o MH and TULMGAT w/o GAT have the lowest performance in all metrics, illustrating that the graph module plays the most important role in TULMGAT. Interestingly, GAT with the single-headed attention mechanism cannot perform effective node representation learning because TULMGAT w/o MH is far inferior to TULMGAT w/o GAT in various metrics. Comparing TULMGAT with TULMGAT w/o BiLSTM, TULMGAT w/o LSTM, TULMGAT w/o MS & LSTM exhibit that BiLSTM in trajectory embedding module plays an active role in this model and simpler trajectory embedding approach is unacceptable. Observing the performance of TULMGAT and TULMGAT-w/o MH, TULMGAT was generally superior to TULMGAT-w/o MH, but the differences were not significant in each metric. According to the results of TULMGAT w/o PG, we find a small decrease in model effectiveness after removing the geographic information from pre-defined graphs. We infer that pre-defined graphs provide useful geographic information and alleviate data sparsity [27], but the network structure is more important.

Overall, the ablation study shows that the graph attention network module plays an irreplaceable role in TULMGAT, the RNN-based trajectory embedding module and the rule-based pre-defined graph have positive effects on mining human mobility patterns, and the multi-scale trajectory sampling module brings limited improvement. In the next section, we will compare TULMGAT with TULGAT to investigate the effectiveness of multi-scale trajectory sampling from a deeper perspective.

5.3.5. Parameter analysis

In this section, we examine each important parameter in TULMGAT and compare it with TULGAT in which the trajectory will generate a single completed *COG* for subsequent modules rather than being sampled at multiple scales to measure the effectiveness of the multi-scale trajectory sampling module.

As shown in Fig. 9, we examine the performance of TULMGAT and TULGAT on Gowalla |247| and Brightkite |223| with different node embedding sizes. It can be intuitively seen that the larger the node embedding dimension is, the better the model performance is. However, when the node embedding size is larger than 128, the model performance no longer significantly improves. Moreover, TULMGAT generally outperforms TULGAT in all metrics.

Next, we investigate the effect of the dropout rate in the GAT module on the performance of the model. As shown in Fig. 10, we examined the dropout rate from 0 to 0.8 on both models. Obviously, as the dropout rate increases, each metric of the model first rises slowly, and when the dropout rate exceeds 0.25, the decreasing trend of the model's performance gradually accelerates. In addition, the metrics of TULMGAT are generally better than those of TULGAT, and its performance degradation rate is remarkably lower than that of TULGAT with the increase in dropout rate.

The number of self-attention heads K in Eq. (4) is an important parameter for the model TULMGAT. The performances of TULMGAT and TULGAT by varying K from 1 to 12 is reported in Fig. 11. Apparently, the effect of the model gradually rises as the number of heads increases, and after $K \geq 10$, the improvement in performance is no longer evident. Moreover, the performance of TULGAT is unsurprisingly lower than that of TULMGAT, and the performance of the model improves slowly when the number of heads is small.

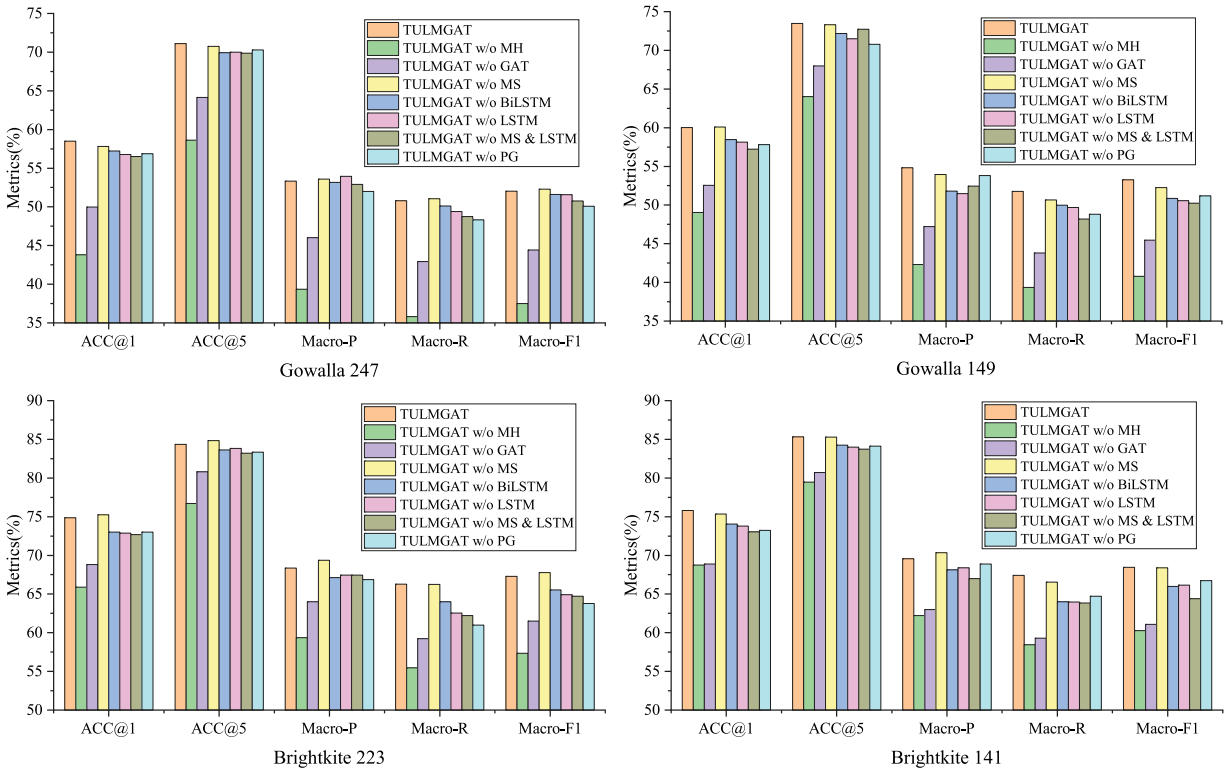

Fig. 8. Ablation study.

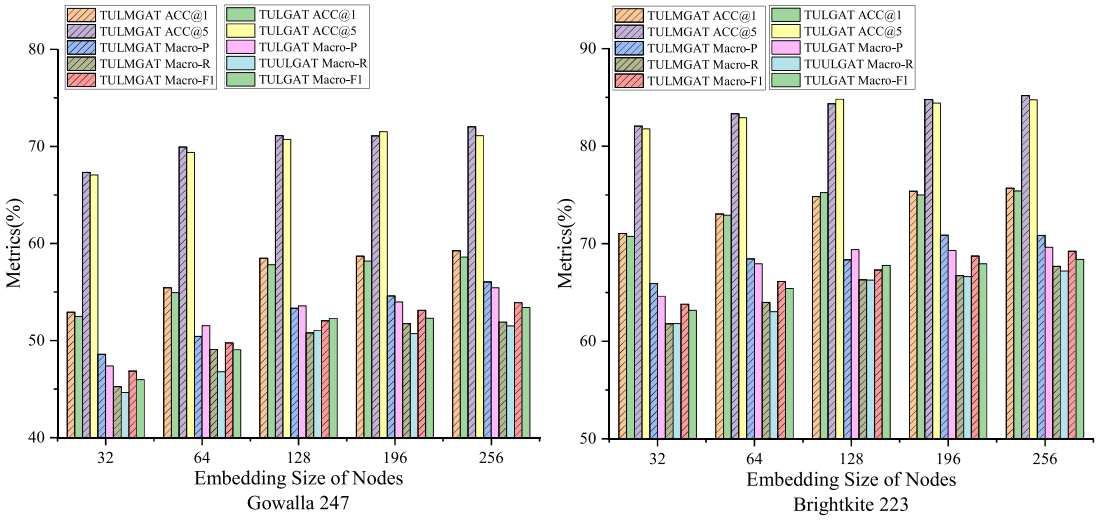

Fig. 9. Performance of TULMGAT and TULGAT on various metrics with different node embedding sizes.

In summary, we analyze the influence of the main modules and parameters of the model on the validity of the model. Moreover, we compare TULMGAT with TULGAT on this basis. Through the analysis, we can find that complete retention of trajectory information is important by examining TULGAT in the single-scale experiment. However, this does not mean that sampling the trajectory at a single-scale is the most efficient. By comparing with TULMGAT at three sampling scales, we find that the performance of TULGAT at a single sampling scale is unstable, and it is more affected by various main parameters such as dropout rate than TULMGAT. When the dropout rate increases or the number of heads decreases, TULGAT performs poorly. This implies that TULGAT is not robust but severely affected by the parameters, which reflects the role of our multiscale sampling module in constructing the *COG*. TULMGAT is less affected by parameter changes, demonstrating that it can perform more robustly in different datasets and parameters.

6. Conclusion and future work

In this paper, a novel model, namely TULMGAT, is developed for the task TUL based on graph attention networks. Specifically, we first construct a single trajectory into several check-in oriented graphs according to different sampling scales and then update the nodes in the trajectory with a masked multi-head self-attention graph neural network to mine the spatio-temporal properties of user mobility in the trajectory. Next, we propose global average pooling to converge the trajectory representations from the graph module and apply the BiLSTM model to learn the interactions of each dimension of the trajectory

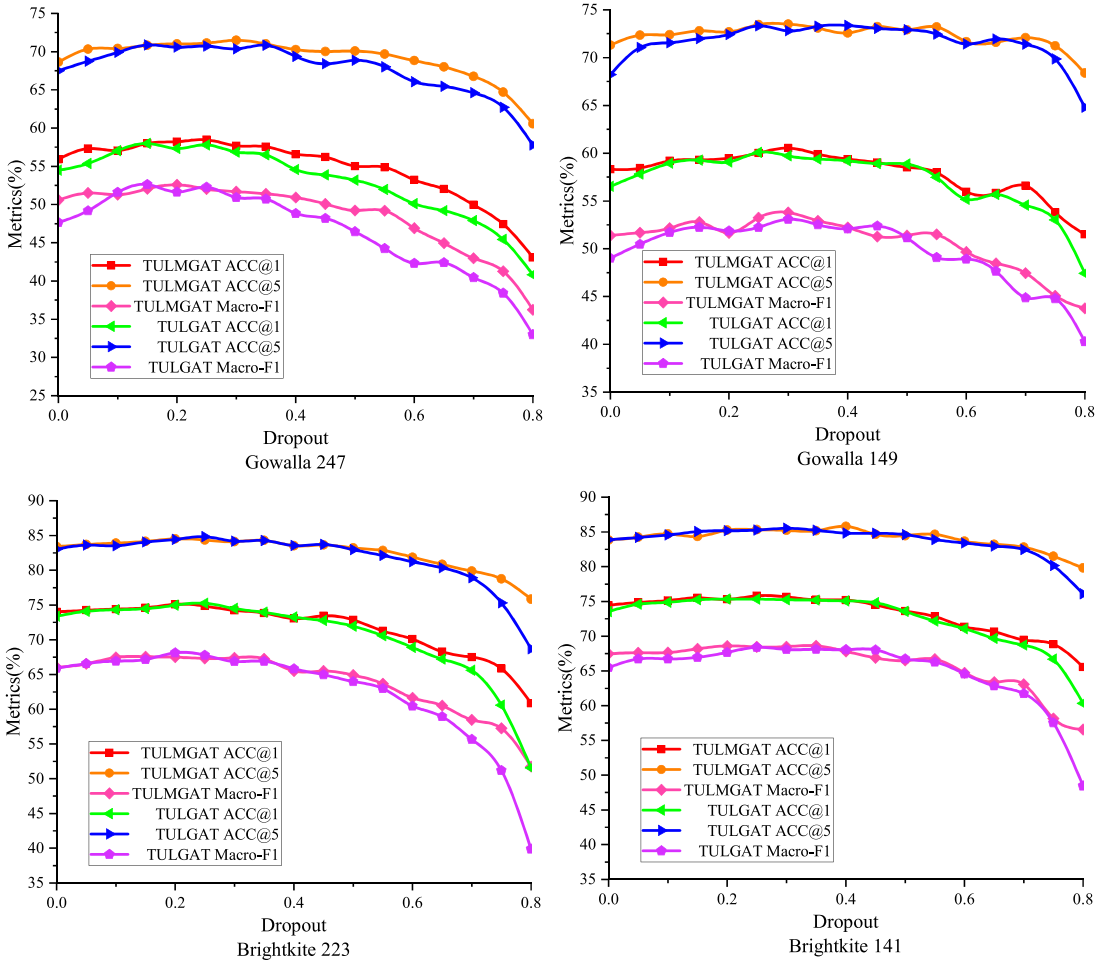

Fig. 10. Performance of TULMGAT and TULGAT on various metrics with different dropout rates.

vectors. Finally, a linking module based on MLP and softmax functions is employed to link trajectories to users. The adequate experiments conducted on two public datasets demonstrate the superiority of our proposed model.

While our work has made many improvements in Trajectory-User Linking, there are still many problems:

- **Selection of grid granularity:** Although Grid index does not require additional information or cumbersome pre-processing methods as POIs, grid granularity needs to be explored when dealing with different datasets. Since our work only applies check-in data from social platforms, the grid granularity is a uniform one-thousandth of one degree of latitude and longitude, which is approximately 100 m. However, in urban datasets such as GeoLife [34], the granularity might be one meter.
- **Inequity in the representation of spatial information:** As shown in Section 4.1.1, Grid Index in latitude and longitude division is actually not fair, one degree of longitude constant equals to 111 kilometers, but one degree of latitude equals to $111 \times cos\theta$ kilometers, where θ is the latitude. Moreover, Grid Index may not be as effective as a non-linear representation of POIs for representing spatial information in complex scenes, even though it seems to work well on check-in data so far.
- **Temporal information:** We have not explored the temporal information of the trajectories in depth, and since we wish to deal with the full information of trajectories with arbitrary length, we have not been able to model the temporal information effectively as in the previous work. The timing information of the trajectories is only applied in our work to establish the edge relationships of nodes, and RNN-based model in Trajectory embedding is just utilized to optimize the representation of trajectories as well as the relationships between multi-scale graphs.
- **Spatial information:** We think that our graph network is just a preliminary work, and there is still much room for exploration of trajectory which is a kind of graph, such as multi-graph construction under different granularity of grids in CULVAE [27], node encoding with timestamp information in time series prediction [35], graph learning [36] and so on.
- **Location semantics:** A wider range of spatial tasks reveal the importance of location semantics. Since a user's trajectory is more than pure spatial movement, each geographic location may contain specific semantic information, such as administrative divisions or urban functional areas. Existing work on location semantics [37] basically guarantees that the introduction of this information will be beneficial and guides an exciting direction of improvement in trajectory user linking.

In future, we will pay more attention to the efficient representation of temporal information in the trajectory since the temporal order in TULMGAT only provides directed edges in trajectories. We will explore the efficient encoding of timestamps or the construction of spatio-temporal trajectory-oriented graphs based on periodic time intervals, and we will perform unsupervised learning to determine the transportation mode based on the speed of user movement for targeted classification of trajectories. Moreover, we will explore more effective spatio-temporal semantic information fusion mechanisms specifically for trajectory or check-in data. Ultimately, we expect to achieve more

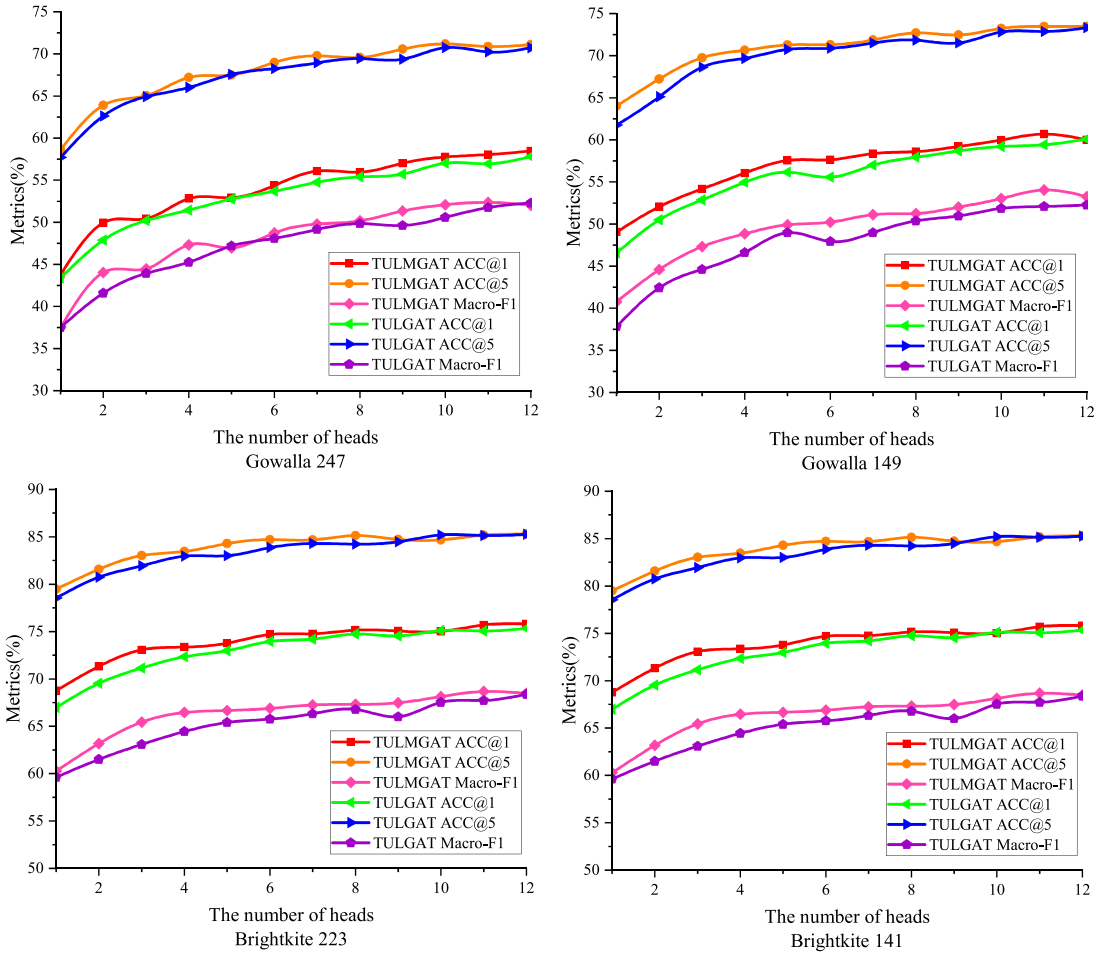

Fig. 11. Performance of TULMGAT and TULGAT on various metrics with different K.

generalized and more effective learning of human mobility patterns in the complex urban traffic environment by studying spatio-temporal data such as trajectories.

CRediT authorship contribution statement

Yujie Li: Writing – review & editing, Writing – original draft, Visualization, Validation, Formal analysis, Data curation, Conceptualization. **Tao Sun:** Writing – review & editing, Validation, Supervision, Project administration. **Zezhi Shao:** Writing – review & editing, Validation, Supervision, Methodology. **Yiqiang Zhen:** Writing – review & editing, Software, Formal analysis, Data curation. **Yongjun Xu:** Resources, Project administration, Investigation, Funding acquisition. **Fei Wang:** Supervision, Resources, Project administration, Investigation, Funding acquisition, Data curation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This work is supported by NSFC, China No. 62372430 and the Youth Innovation Promotion Association CAS No. 2023112. We thank all the anonymous reviewers who generously contributed their time and efforts. Their professional recommendations have greatly enhanced the quality of the manuscript.

References

[1] Y. Ma, S. Liu, G. Xue, D. Gong, Soft sensor with deep learning for functional region detection in urban environments, Sensors 20 (12) (2020) 3348.
[2] C. Wang, C. Liang, X. Chen, H. Wang, Identifying effective trajectory predictions under the guidance of trajectory anomaly detection model, Pattern Recognit. 140 (2023) 109559.
[3] J. Li, L. Yang, Y. Chen, Y. Jin, MFAN: Mixing feature attention network for trajectory prediction, Pattern Recognit. 146 (2024) 109997.
[4] Q. Gao, F. Zhou, K. Zhang, G. Trajcevski, X. Luo, F. Zhang, Identifying human mobility via trajectory embeddings, in: IJCAI, Vol. 17, 2017, pp. 1689–1695.
[5] F. Wang, D. Yao, Y. Li, T. Sun, Z. Zhang, AI-enhanced spatial-temporal data-mining technology: New chance for next-generation urban computing, Innovation 4 (2) (2023).
[6] C. Liu, Y. Wang, D. Li, X. Wang, Domain-incremental learning without forgetting based on random vector functional link networks, Pattern Recognit. 151 (2024) 110430.
[7] H. Cao, H. Tang, F. Wang, Y. Xu, Survey on trajectory representation learning techniques, J. Softw. 32 (5) (2021) 1461–1479.
[8] Y. Xu, X. Liu, X. Cao, C. Huang, E. Liu, S. Qian, X. Liu, Y. Wu, F. Dong, C.-W. Qiu, et al., Artificial intelligence: A powerful paradigm for scientific research, Innovation 2 (4) (2021) 100179.
[9] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, NeurIPS 26 (2013).

[10] P. Yan, Y. Tan, Y. Tai, D. Wu, H. Luo, X. Hao, Unsupervised learning framework for interest point detection and description via properties optimization, Pattern Recognit. 112 (2021) 107808.
[11] F. Zhou, Q. Gao, G. Trajcevski, K. Zhang, T. Zhong, F. Zhang, Trajectory-user linking via variational AutoEncoder, in: IJCAI, 2018, pp. 3212–3218.
[12] F. Zhou, X. Liu, K. Zhang, G. Trajcevski, Toward discriminating and synthesizing motion traces using deep probabilistic generative models, IEEE Trans. Neural Netw. Learn. Syst. 32 (6) (2021) 2401–2414.
[13] D.P. Kingma, S. Mohamed, D.J. Rezende, M. Welling, Semi-supervised learning with deep generative models, in: NeurIPS, 2014, pp. 3581–3589.
[14] T. Sun, Y. Xu, F. Wang, L. Wu, T. Qian, Z. Shao, Trajectory-user link with attention recurrent networks, in: ICPR, 2020, pp. 4589–4596.
[15] F. Zhou, S. Chen, J. Wu, C. Cao, S. Zhang, Trajectory-user linking via graph neural network, in: ICC 2021, 2021, pp. 1–6.
[16] Y. Sang, Z. Xie, W. Chen, L. Zhao, TULRN: Trajectory user linking on road networks, World Wide Web (2022) 1–17.
[17] W.L. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, in: NeurIPS, 2017, pp. 1024–1034.
[18] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, Graph attention networks, in: ICLR, 2018.
[19] F. Zhou, R. Yin, G. Trajcevski, K. Zhang, J. Wu, A.A. Khokhar, Improving human mobility identification with trajectory augmentation, GeoInformatica 25 (3) (2021) 453–483.
[20] C. Miao, J. Wang, H. Yu, W. Zhang, Y. Qi, Trajectory-user linking with attentive recurrent network, in: AAMAS, 2020, pp. 878–886.
[21] F.T. Islam, M.T. Mahmood, M. Naznin, MTUL: a novel approach for multi-trajectory user linking, in: NSysS, 2022, pp. 83–91.
[22] W. Chen, S. Li, C. Huang, Y. Yu, Y. Jiang, J. Dong, Mutual distillation learning network for trajectory-user linking, in: IJCAI, 2022, pp. 1973–1979.
[23] S. Zhang, S. Wang, X. Wang, S. Zhang, H. Miao, J. Zhu, Multi-task adversarial learning for semi-supervised trajectory-user linking, in: ECML PKDD.
[24] C. Yu, F. Wang, Z. Shao, T. Qian, Z. Zhang, W. Wei, Y. Xu, GinAR: An end-to-end multivariate time series forecasting model suitable for variable missing, in: KDD, 2024, pp. 3989–4000.
[25] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: ICLR, 2017.
[26] Q. Sang, Z. Lin, S.T. Acton, A grid-based tracker for erratic targets, Pattern Recognit. 48 (11) (2015) 3527–3541.
[27] Y. Li, Y. Sang, W. Chen, L. Zhao, Linking check-in data to users on location-aware social networks, in: PRICAI 2022, Vol. 13629, 2022, pp. 489–503.
[28] T. Sun, Y. Xu, Z. Zhang, L. Wu, F. Wang, A hierarchical spatial-temporal embedding method based on enhanced trajectory features for ship type classification, Sensors 22 (3) (2022) 711.
[29] X. Liu, Y. Xia, Y. Liang, J. Hu, Y. Wang, L. Bai, C. Huang, Z. Liu, B. Hooi, R. Zimmermann, LargeST: A benchmark dataset for large-scale traffic forecasting, in: NeurIPS 2023.
[30] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, A. Alahi, Social GAN: socially acceptable trajectories with generative adversarial networks, in: CVPR, 2018, pp. 2255–2264.
[31] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, Q. Mei, LINE: large-scale information network embedding, in: WWW, 2015, pp. 1067–1077.
[32] C. Zhang, K. Zhang, Q. Yuan, H. Peng, Y. Zheng, T. Hanratty, S. Wang, J. Han, Regions, periods, activities: Uncovering urban dynamics via cross-modal representation learning, in: WWW, ACM, 2017, pp. 361–370.
[33] A. Najjar, K. Mede, Trajectory-user linking is easier than you think, in: IEEE BigData, 2022, pp. 4936–4943.
[34] W. Chen, C. Huang, Y. Yu, Y. Jiang, J. Dong, Trajectory-user linking via hierarchical spatio-temporal attention networks, TKDD 18 (4) (2024) 1–22.
[35] Z. Shao, F. Wang, Y. Xu, W. Wei, C. Yu, Z. Zhang, D. Yao, G. Jin, X. Cao, G. Cong, et al., Exploring progress in multivariate time series forecasting: Comprehensive benchmarking and heterogeneity analysis, 2023, arXiv preprint arXiv:2310.06119.
[36] Y. Li, Z. Shao, Y. Xu, Q. Qiu, Z. Cao, F. Wang, Dynamic frequency domain graph convolutional network for traffic forecasting, in: 2024 IEEE International Conference on Acoustics, Speech and Signal Processing, 2024, pp. 5245–5249.
[37] C. Chen, Y. Ren, H. Liu, Y. Chen, H. Li, Acoustic-sensing-based location semantics identification using smartphones, IEEE Internet Things J. 9 (20) (2022) 20640–20650.

Yujie Li received the B.E. degree from Soochow University, Suzhou, China, in 2023. He is currently pursuing a Master's degree with the Institute of Computing Technology, Chinese Academy of Sciences, China. His research interests include spatio-temporal data mining, graph neural networks, deep learning.

Tao Sun received his Ph.D. in computer system structure in 2022 from the University of Chinese Academy of Sciences. He is currently an assistant professor at the Institute of Computing Technology, Chinese Academy of Sciences. His research interest is in the area of spatial–temporal data mining.

Zezhi Shao received the B.E. degree from Shandong University, Jinan, China, in 2019. He is currently pursuing a Ph.D. degree with the Institute of Computing Technology, Chinese Academy of Sciences, China. His research interests include multivariate time series forecasting, graph neural network, and data mining.

Yiqiang Zhen received his M.S. degree in 2006 from Beijing Institute of Tracking and Communication Technology. He is currently working in DFH Satellite Co., Ltd. and his main research areas include satellite applications, data processing and data mining.

Yongjun Xu is a professor at Institute of Computing Technology, Chinese Academy of Sciences (ICT-CAS) in Beijing, China. He received his Ph.D. degree in Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China in 2006. His current research interests include artificial intelligence systems, and big data processing.

Fei Wang received the Ph.D. degree in computer architecture from University of Chinese Academy of Sciences in 2017. Since 2020, he has been working as associate professor in Institute of Computing Technology, Chinese Academy of Sciences. His main research interest include spatiotemporal data mining, information fusion, graph neural networks.