Representation Learning of Temporal Graphs with Structural Roles

Huaming Du
School of Business Administration
Southwestern University of Finance and Economics
Chengdu, Sichuan, China
dhmfcc@smail.swufe.edu.cn

Long Shi
Financial Intelligence and Financial Engineering Key Laboratory
Southwestern University of Finance and Economics
Chengdu, Sichuan, China

Xingyan Chen
Financial Intelligence and Financial Engineering Key Laboratory
Southwestern University of Finance and Economics
Chengdu, Sichuan, China

Yu Zhao*
Financial Intelligence and Financial Engineering Key Laboratory
Southwestern University of Finance and Economics
Chengdu, Sichuan, China

Hegui Zhang†
School of Data Science and Artificial Intelligence
Dongbei University of Finance and Economics
Dalian, Liaoning, China

Carl Yang
Department of Computer Science
Emory University
Atlanta, Georgia, United States

Fuzhen Zhuang
Institute of Artificial Intelligence
SKLSDE, School of Computer Science,
Beihang University
Beijing, China

Gang Kou
Xiangjiang Laboratory
Changsha, Hunan, China
School of Business Administration
Southwestern University of Finance and Economics
Chengdu, Sichuan, China

ABSTRACT

Temporal graph representation learning has drawn considerable attention in recent years. Most existing works mainly focus on modeling local structural dependencies of temporal graphs. However, underestimating the inherent global structural role information in many real-world temporal graphs inevitably leads to suboptimal graph representations. To overcome this shortcoming, we propose a novel **R**ole-based **T**emporal **G**raph **C**onvolution **N**etwork (**RTGCN**) that fully leverages the global structural role information in temporal graphs. Specifically, RTGCN can effectively capture the static global structural roles by using hypergraph convolution neural networks. To capture the evolution of nodes' structural roles, we further design structural role-based gated recurrent units. Finally, we integrate structural role proximity in our objective function to preserve global structural similarity, further promoting temporal graph representation learning. Experimental results on multiple real-world datasets demonstrate that RTGCN consistently outperforms state-of-the-art temporal graph representation learning methods by significant margins in various temporal link prediction and node classification tasks. Specifically, RTGCN achieves AUC improvement of up to 5.1% for link prediction and F1 improvement of up to 6.2% for new link prediction. In addition, RTGCN achieves AUC improvement up to 4.6% for node classification and 2.7% for structural role classification.

CCS CONCEPTS

• **Information systems** → **Social networks**; • **Computing methodologies** → **Learning latent representations**.

KEYWORDS

Temporal graph representation learning, Structural roles

ACM Reference Format:
Huaming Du, Long Shi, Xingyan Chen, Yu Zhao, Hegui Zhang, Carl Yang, Fuzhen Zhuang, and Gang Kou. 2024. Representation Learning of Temporal Graphs with Structural Roles. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24), August 25–29, 2024, Barcelona, Spain.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3637528.3671854

1 INTRODUCTION

Temporal graph representation learning is important since it captures how the node interacts and the graph evolves, which will help to understand and predict events in complex dynamic systems, for instance, the variation of traffic flows in transportation networks [8] and researchers' interest shifts in academic networks [31]. Early

*co-corresponding author
†corresponding author (hegui.zhang@dufe.edu.cn; hegui.zhang@qq.com)

KDD'2024, Aug 25–29, 2024, Barcelona, Spain

ACM ISBN 979-8-4007-0490-1/24/08
https://doi.org/10.1145/3637528.3671854

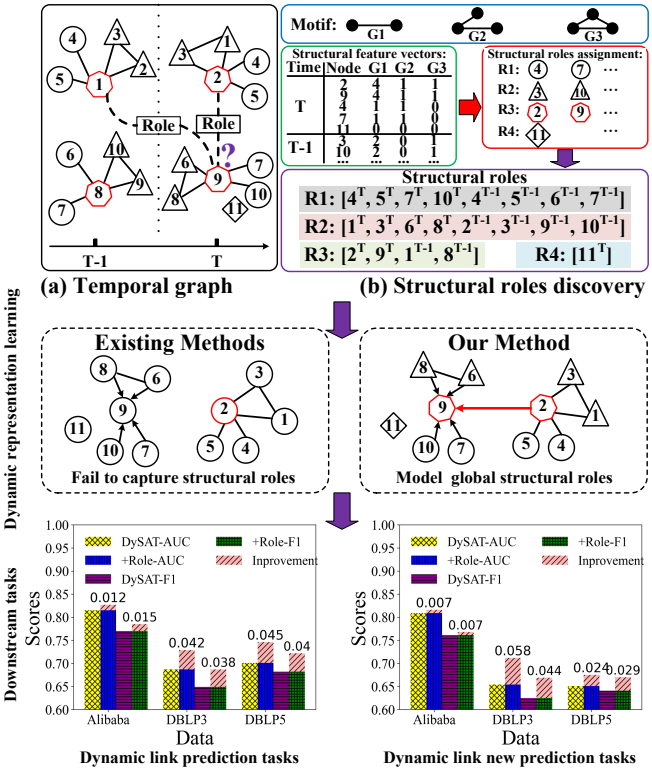


Figure 1: The effectiveness of structural roles for temporal graph learning in various downstream tasks. (a) Temporal graph: Nodes in the temporal graph are labeled. Superscripts on nodes indicate timesteps. (b) Structural roles discovery: Firstly, we construct the structural feature vector for each node. Then, we assess the structural similarity between nodes based on the structural feature vectors. Lastly, we assign structurally similar nodes to the same structural role set (i.e., R1, R2, R3, R4).

temporal graph methods adopt matrix decomposition to capture graph structures in each snapshot and regularize the smoothness of the representations of adjacent snapshots [9]. Such matrix decomposition is usually computationally heavy. With the development of deep learning, graph neural networks [7] are adopted to capture the structural information while recurrent neural networks or transformer [26] are further utilized to summarize the temporal information [17, 22, 32]. Despite achieving preliminary successes, existing methods typically generate node representations through local connectivity proximity[1], neglecting global structural similarity[2].

Recently, in the domains of complex networks and static graphs, it has been demonstrated that modeling structural roles of nodes is a crucial step toward better representation learning [1, 6, 35]. Intuitively, *if two nodes share similar neighborhood connectivity patterns, they are structurally similar, and thus, they belong to the same structural role.* As shown in Figure 1, △ (R3) describes a connectivity pattern of a triangle. This definition allows nodes to be neither directly connected nor even in the same snapshot. Therefore, structural roles have the capability to model *long-range dependencies*, capturing crucial information from distant nodes. Furthermore, structural roles can model *global dependencies* and capture all crucial node information in the entire graph or even within the entire snapshots to enhance the learning of temporal graph representations. As shown in Figure 1(a), the goal is to predict the label of target node 9 in snapshot T. With structural role information, we can directly utilize non-adjacent node information (e.g., node 2^T and node 1^{T-1}) at the current snapshot and historical snapshots to accurately predict the label of node 9. However, existing methods lack the ability to capture long-range dependencies in temporal graphs, thus failing to capture essential features from distant but informative nodes [18]. This limitation may significantly restrict the representational capacity (bottom of Figure 1).

Modeling the evolving patterns of temporal graphs becomes a crucial research question [10, 32, 34]. Initially, we predefine structural roles based on the temporal graph and then assigning role definitions to each node in different snapshots. As the temporal graph evolves, the connection patterns of nodes change, leading to variations in node structural roles. Through the evolution of structural roles, we can effectively characterize the evolving patterns of the temporal graph. Importantly, we do not alter the definition of structural roles in this process. For example, at time $T-1$, node 8 transitions from structural role R2 to R3 at time T. This transformation essentially denotes the change in R2 and R3 structures from snapshot $T-1$ to snapshot T, capturing a global evolution pattern. In contrast, traditional temporal graph methods using RNNs, transformers [26], or SNNs [10] can only model changes in the local connection patterns of nodes. As illustrated in Figure 1, existing methods can only depict the transition of node 8's neighbors at time $T-1$, changing from the neighbor set [6, 7, 9, 10] to [6, 9] at time T.

Due to the temporal and structural complexity in temporal graphs, considering structural role information is not a simple task of extension, it poses the following *three challenges*: (1) How to efficiently model the interactions between nodes within the same structural role set in same snapshot? (2) How to consider the influence of nodes with the same structural role from previous snapshots on the current snapshot? (3) How to better utilize structural role information to model the evolution patterns of temporal graphs effectively?

To address these challenges, we propose a novel structural role-based temporal graph convolutional network, namely, the **RTGCN**, to preserve both local connective proximity and global structural similarity under a unified framework, focusing on capturing the global structural roles and their evolutions. Specifically, in RTGCN, we first use structural role information to construct hypergraphs for the current snapshot and historical snapshots. Furthermore, we propose a structural role-based Graph Neural Network (GNN) module to capture the global structural similarity of the nodes. Moreover, we design a structural role-based Gated Recurrent Unit (GRU) along with the temporal dimension to effectively capture the evolving patterns of the structural roles in the temporal graphs.

[1]If two nodes are connected by edges or paths, the node embeddings should be similar.
[2]If two nodes are not connected but exhibit similar neighborhood structures, their embeddings should also be similar.

Finally, a constraint based on structural role similarity is introduced to further preserve global structural similarity.

In summary, the main contributions are stated as follows:

• To the best of our knowledge, we are the first to incorporate global structural role information into the representation learning of temporal graphs.

• Our proposed RTGCN applies a hypergraph-based GNN module to model the structural roles and a role-based GRU module to capture the evolution of structural roles in temporal graphs.

• Extensive experimental results on diverse real-world temporal graphs demonstrate the superiority of RTGCN as it consistently outperforms 15 SOTA baselines on all datasets for various tasks.

2 RELATED WORK

Temporal graph representation learning. It has been widely studied in recent years. Most approaches integrate GNN with the recurrent architecture, whereby the former captures graph structure information and the latter handles temporal dynamism via maintaining a hidden state to summarize historical snapshots [12, 13, 17, 22–24]. For instance, ROLAND [32] repurposes static GNN designs to temporal graph settings and proposes a live-update evaluation setting to learn node representations. HTGN [29, 30] studies temporal graph representation learning built upon a hyperbolic geometry powered with the recurrent learning paradigm. Recently, SpikeNet [10] captures the evolving dynamics of temporal graphs with spiking neural networks via the LIF [4] model. While the methods discussed all intend to capture temporal evolving patterns, none of them can capture the global structural role information in temporal graphs.

Role-based embedding methods. They mainly focus on the research of single-layer networks, and few research is extended to multiplex networks. For role-based single-layer network embedding, some methods implement role assignment of nodes through the role discovery process and then improve the random walk method to learn node representations [1, 14]. Other methods learn role-based embedding based on feature decomposition and utilize graph kernels to capture the structural similarity between nodes [16, 21]. The embedding methods that can capture the similarity between nodes can also be classified as role-oriented embedding methods [6, 19, 20]. For multiplex networks, RMNE [33] derives role-based embedding from extending role-aware random walks or incorporating structural role information into a unified learning framework. These methods do not consider the dynamic changes of the graph structure and node features, rendering them unsuitable for direct application in temporal graph representation learning.

3 METHODOLOGY

As shown in Figure 2, RTGCN is a recurrent learning paradigm, fitting within the prevalent discrete-time temporal graph architecture. The RTGCN unit primarily consists of three components: (1) Structural Role Hypergraph Construction, the module aims to construct hypergraphs to extract global structural role information; (2) Structural Role-based GRU Module, the modified temporal recurrent construct designed to capture the sequential patterns of nodes; (3) Structural Role-based GNN Module, the graph and hypergraph neural network to model global structure role information. We elaborate on the details of each module in the following paragraphs. Prior to this exposition, we initially present the formal definition of structural roles.

Structural Roles. Given a graph $G(V, E)$, we first extract the structural feature vector $F = [f_1, f_2, ..., f_m]$ of each node. Then, each node is mapped into a set of structural roles through a mapping function Φ. We achieve this by manually defining a function that maps the $|V|$ vertices to a set $W = [\omega_1, \omega_2, ..., \omega_M]$ of M structural roles sets where $M \leqslant |V|$. Let u and v are arbitrary nodes, the specific mapping process is as follows:

$$\Phi : (\forall i, 1 \leqslant i \leqslant m : f_i(u) = f_i(v)) \Rightarrow \Phi(u) \equiv \Phi(v) \tag{1}$$

Please note that the definition of structural roles we propose is flexible and can employ various mapping functions.

3.1 Structural Role Hypergraph Construction

To extract global structural role information, this module focuses on constructing structural role-based hypergraph incidence matrices $\boldsymbol{H}$, both for the current and historical snapshots:

$$\boldsymbol{H}(v, \boldsymbol{e}) = \begin{cases} \frac{1}{exp(w\Delta t)}, & if\ v \in \boldsymbol{e} \\ 0, & if\ v \notin \boldsymbol{e} \end{cases} \tag{2}$$

where $\boldsymbol{e}$ is the hyperedge (i.e., structural role set), w is the time decay factor and Δt denotes the time interval from the current snapshot.

The primary objective of our method is to leverage structural role information to augment the representation learning process in temporal graphs. Consequently, the contribution of structural role information at the current snapshot and historical snapshots to node representation learning may be different. To address this, structural role hypergraphs are constructed for both the current snapshot and the historical snapshots, to enable subsequent modules (i.e., structural role-based GRU) to effectively differentiate between them. Next, we will describe the specific construction process.

3.1.1 Hypergraph Construction of Current snapshot. We use the role incidence matrix to capture the structural role information at the current snapshot. As previously described, nodes in snapshot t are classified within the same structural role set if they exhibit structural similarity. It's a natural approach to connect these nodes, enabling them to disseminate and aggregate messages. However, this method falters in capturing global topological relationships within the same structural role set due to over-smoothing as the number of GCN layers increases. In contrast, the hypergraph is more effective in learning node and graph representation. Given the $\Delta t = 0$, the hypergraph incidence matrix $\boldsymbol{H_{cu}} \in \mathbb{R}^{|V| \times |\varepsilon|}$ of current snapshot can be obtained from Eq. (2). Please note that $|\varepsilon|$ is significantly smaller than the number of nodes N.

3.1.2 Hypergraph Construction of historical snapshots. Historical information plays an indispensable role in temporal graph modeling since it facilitates the model to learn the evolving patterns and regularities. However, traditional recurrent neural networks can not fully capture some complex evolving patterns due to their monotonic mechanism and inherent limitations [11].

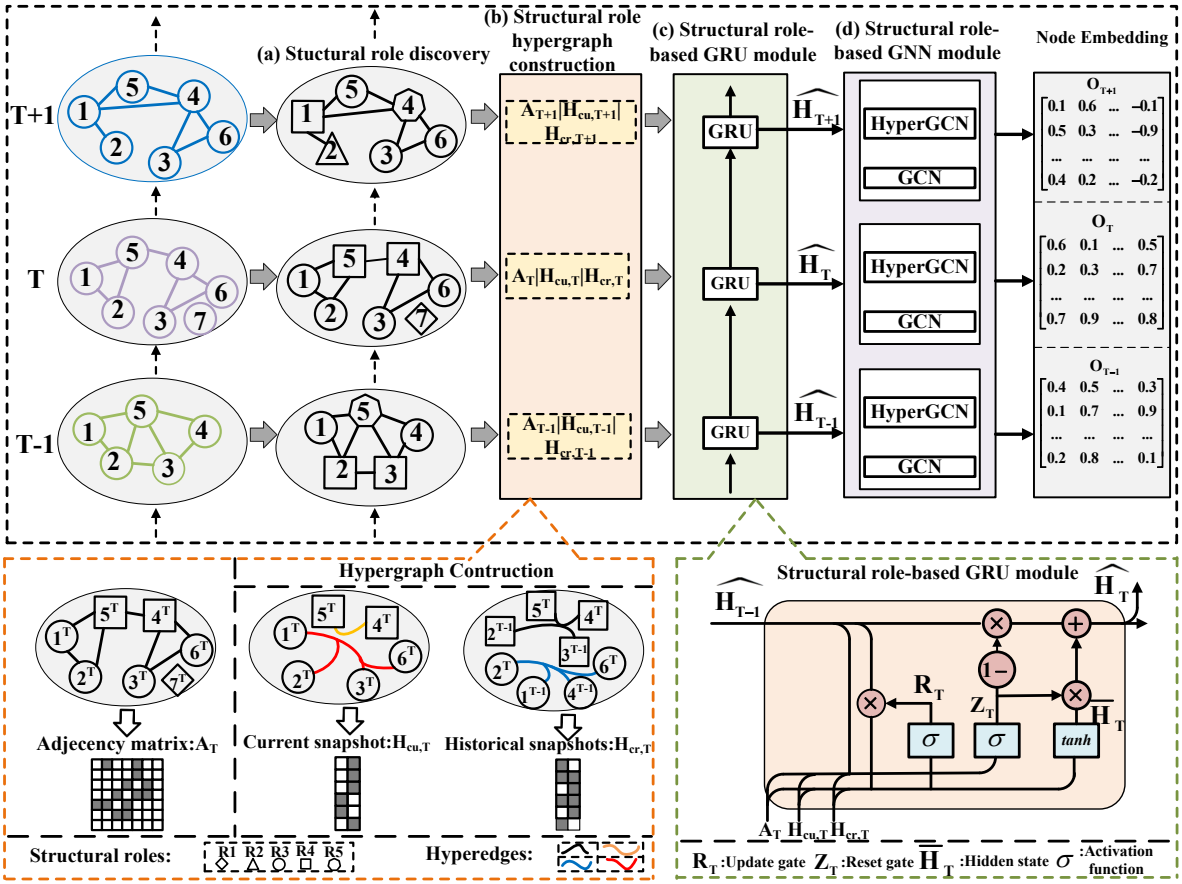


Figure 2: An overview of RTGCN. (a) Structural role discovery: Construct structural role features for each node and assign roles to each node based on these structural role features. (b) Structural Role hypergraph construction: Nodes with the same role are connected through hyperedges, constructing hypergraphs for the current snapshot and historical snapshots based on role information. The specific construction details are illustrated within the *orange dashed box*. (c) Structural Role-based GRU Module: Utilizes the previously constructed hypergraph structural information as input to the GRU module to regulate the parameters of the GNN, thereby modeling the complex evolution of the temporal graph. *Green dashed box* represents specific details. (d) Structural Role-based GNN Module: This module comprises two hypergraph convolutional neural networks and one conventional graph convolutional neural network. Eventually, the obtained nodes' comprehensive representations are utilized for various downstream tasks. Please note that, in the construction of hypergraphs for historical snapshots, we provide an example by considering only the influence of nodes in snapshot $T-1$ on the nodes in current snapshot T.

Inspired by [15, 29], we assume that nodes of the same role in the previous k snapshots and at the current snapshot t can conduct message propagation and aggregation, attending to multiple historical states to describe the evolving patterns of nodes. In addition, we introduce a time decay function to measure the attenuation of the influence effect. When $\Delta t > 0$, we can acquire the hypergraph incidence matrix $\boldsymbol{H}_{cr}$ of historical snapshots from the Eq. (2).

3.2 Structural Role-based GRU Module

Traditional methods for modeling the graph evolution process can be divided into two categories. One utilizes a sequence model that only builds on the output embedding of GNN to keep and update node state [10, 12, 32], and the other uses the output of GNN and adjacency matrix to update learnable parameter of GNN [24].

Unlike traditional methods, when designing the temporal module, we mainly consider the following three aspects: First, these existing methods fail to effectively capture the dynamics of the graph, as discussed in Section 1. Second, along the temporal axis, the architecture of the model remains consistent, and therefore, we only need to update the model parameter matrix. Lastly, in practical applications, nodes frequently appear and disappear, making node embedding-based sequence methods questionable. Inspired by existing research [17], to efficiently capture the complex evolution process of the graph, we utilize a modified-GRU to model the temporal dependency among learnable parameters of RTGCN at different snapshots. Here, based on the modified GRU, we use the global structural role information to update the model weight parameter at each snapshot. Our subsequent experiments further validate the

effectiveness of this module, as shown in Figures 9 in Appendix B.1. Formally, we take the $\boldsymbol{A}$, $\boldsymbol{H_{cu}}$ and $\boldsymbol{H_{cr}}$ as the input of GRU to generate the weight matrix of the next snapshot, which is formulated as:

$$Z_t = \sigma\left(W_{z_1} A_t W_{z_{11}} + \left(W_{z_2} H_{cu,t} + W_{z_3} H_{cr,t}\right) W_{z_{12}} + U_z \widehat{H_{t-1}}\right), \tag{3a}$$

$$R_t = \sigma\left(W_{r_1} A_t W_{r_{11}} + \left(W_{r_2} H_{cu,t} + W_{r_3} H_{cr,t}\right) W_{r_{12}} + U_r \widehat{H_{t-1}}\right), \tag{3b}$$

$$\widetilde{H}_t = tanh\left(W_1 A_t W_{11} + \left(W_2 H_{cu,t} + W_3 H_{cr,t}\right) W_{12} + U\left(R_t \circ \widehat{H_{t-1}}\right)\right), \tag{3c}$$

$$\widehat{H_t} = (1 - Z_t) \circ \widehat{H_{t-1}} + Z_t \circ \widetilde{H}_t, \tag{3d}$$

where $\boldsymbol{A_t}$ is the adjacency matrix, and $\widehat{\boldsymbol{H_{t-1}}}$ represents the weight matrix of last snapshot. Note that, $\widehat{\boldsymbol{H_t}}$ consists of $\boldsymbol{\Theta}_{cu,t}$, $\boldsymbol{\Theta}_{cr,t}$, and $\boldsymbol{W}_{a,t}$, which can be shared in some downstream tasks. $\boldsymbol{\Theta}_{cu,t}$, $\boldsymbol{\Theta}_{cr,t}$, and $\boldsymbol{W}_{a,t}$ represent the learnable parameters corresponding to $\boldsymbol{H}_{cu}$,$\boldsymbol{H}_{cr}$, and $\boldsymbol{A}$, respectively. Please note that if there are no initial node features in the dataset or if the dimension of initial node features is the same as the number of nodes, the additional learnable parameter matrix $\boldsymbol{W}_{11}$, $\boldsymbol{W}_{12}$, $\boldsymbol{W}_{z_{11}}$, $\boldsymbol{W}_{z_{12}}$, $\boldsymbol{W}_{r_{11}}$, and $\boldsymbol{W}_{r_{12}}$ can be ignored.

3.3 Structural Role-based GNN Module

The Structural Role-based GNN module is employed to model global structure role information in temporal graphs. Beyond the inherent pairwise graph structures present in the adjacency matrix, we encompass two previously discussed hypergraph structures to model the global structural similarity of temporal graphs. Specifically, we first use the simplified GCN and hypergraph convolution neural networks (hyperGCN) to learn node representations. This is achieved based on the adjacency matrix and the two hypergraph incidence matrices, respectively. Then these node representations are integrated to obtain the final node embedding.

At a specific snapshot $\mathcal{G}_t$, the feature propagation on the original graph is defined as:

$$\begin{aligned} \boldsymbol{m}_{v\to u}^{(l)} &= MSG^{(l)}\left(\boldsymbol{h}_v^{(l-1)}, \boldsymbol{h}_u^{(l-1)}\right), \\ \boldsymbol{h}_u^{(l)} &= AGG^{(l)}\left(\left\{\boldsymbol{m}_{v\to u}^{(l)} \mid v \in \mathcal{N}(u)\right\}, \boldsymbol{h}_u^{(l-1)}\right), \end{aligned} \tag{4}$$

where $\boldsymbol{h}_u^{(l)}$ is the node embedding for $u \in V$ after passing through l layers, $\boldsymbol{h}_u^{(0)} = \boldsymbol{x}_u$. The $\boldsymbol{m}_{v\to u}^{(l)}$ is the message embedding and $\mathcal{N}(u)$ is the neighbors of u. To capture global structure role information, the hyperGCN is employed to learn global structural similarity in current snapshot hypergraph and historical snapshots hypergraph. In the current snapshot hypergraph, the hypergraph incidence matrix and the node feature are fed into the hyperGCN to update node embedding. We can build a hypergraph convolution layer [2] in the following formulation:

$$\boldsymbol{X}_{cu}^{l+1} = \sigma\left(\boldsymbol{D}_v^{-1/2} \boldsymbol{H}_{cu} \boldsymbol{W} \boldsymbol{D}_{\boldsymbol{e}}^{-1} \boldsymbol{H}_{cu}^{\top} \boldsymbol{D}_v^{-1/2} \boldsymbol{X}_{cu}^{l} \boldsymbol{\Theta}_{cu}^{l}\right), \tag{5}$$

where $\boldsymbol{X}_{cu}^{l+1}$ is the signal of hypergraph at l layer, $\boldsymbol{X}_{cu}^{0}$ denotes the node initial embedding, $\boldsymbol{\Theta}_{cu}^{l}$ represents learnable parameter of l layer, and $\sigma(\cdot)$ is the activation function. $\boldsymbol{W}$ is the hyperedges weight matrix and we assign each hyperedge the same weight 1. Further, $\boldsymbol{D}_v$ and $\boldsymbol{D}_e$ denote the diagonal matrices of the vertex degrees and the edge degrees, respectively. To model the dependency of nodes of the same structural role in previous snapshots on nodes in the current snapshot, the hyperGCN can also be utilized to learn global structural similarity in the historical snapshots hypergraph. We perform the hypergraph convolution on $\boldsymbol{H}_{cr}$, which is formulated as:

$$\boldsymbol{X}_{cr}^{l+1} = \sigma\left(\boldsymbol{D}_{v1}^{-1/2} \boldsymbol{H}_{cr} \boldsymbol{W} \boldsymbol{D}_{\boldsymbol{e1}}^{-1} \boldsymbol{H}_{cr}^{\top} \boldsymbol{D}_{v1}^{-1/2} \boldsymbol{X}_{cr}^{l} \boldsymbol{\Theta}_{cr}^{l}\right), \tag{6}$$

where $\boldsymbol{\Theta}_{cr}^{l}$ is the learnable parameter of l layer, and $\boldsymbol{X}_{cr}^{0}$ denotes the node initial embedding. $\boldsymbol{D}_{v1}$ and $\boldsymbol{D}_{\boldsymbol{e1}}$ denote the diagonal matrices of the vertex degrees and the edge degrees in $\boldsymbol{H}_{cr}$, respectively. On the top of the node representations, we can obtain more expressive feature $\mathbf{O}$ via weighted sum. Therefore, the $\mathbf{O_t}$ can be defined as:

$$\mathbf{O_t} = \boldsymbol{H}_t + \alpha \boldsymbol{X}_{cu,t} + \gamma \boldsymbol{X}_{cr,t}, \tag{7}$$

where $\mathbf{O_t}$ is the final node embedding, while α and γ are the hyperparameters.

3.4 Model Learning

To preserve local connective proximity and global structural similarity in temporal graphs, we present the objective function from two aspects: connective proximity and structural role proximity, for training our model. Integrating the above modules, we establish the overall learning procedure as summarized in Algorithm 1 in Appendix A.

3.4.1 Connective Proximity. Inspired by [3, 13], we use the dynamic embedding of a node u in snapshot $\mathcal{G}_t$, $\boldsymbol{O}_{t,u}$, to preserve connective proximity around u. In particular, we use a binary cross-entropy loss at each snapshot to increase the similarities between the node embedding appearing in the same fixed-length random walks:

$$\mathcal{L}_c = \sum_{t=1}^{T} \sum_{u=1}^{n} \left(\begin{array}{l} \sum\limits_{v \in n_u^t} -\log\left(\sigma\left(\left\langle \boldsymbol{O}_{t,u}, \boldsymbol{O}_{t,v} \right\rangle\right)\right) \\ -\beta \sum\limits_{v' \in p_u^t} \log\left(1 - \sigma\left(\left\langle \boldsymbol{O}_{t,u}, \boldsymbol{O}_{t,v'} \right\rangle\right)\right) \end{array} \right), \tag{8}$$

where $<,>$ represent any vector similarity measure function (such as the inner product operation), and n_u^t denotes the set of nodes that co-occur with u during a fixed-length random walk. The term p_u^t refers to a negative sampling distribution, typically a function of the node degrees, while β signifies the negative sampling ratio. This ratio acts as an adjustable hyperparameter, providing a mechanism to equilibrate the positive and negative samples within the model.

3.4.2 Structural Role Proximity. Inspired by CTGCN [12], we pose that nodes sharing the same structural role tend to exhibit similar node representation, which enables the RTGCN to preserve global structural similarity. For nodes u and v that align with the same structural role, the concept of structural role proximity is formally defined as follows:

$$\mathcal{L}_r = \frac{1}{n} \sum_{t=1}^{T} \sum_{u=1}^{n} \sum_{v \in R(u)} d\left(\boldsymbol{O}_{t,u}, \boldsymbol{O}_{t,v}\right), \tag{9}$$

where $R(u)$ denotes a set of nodes with the same structural role as the node u, and d can be any distance metric function (e.g., cosine similarity). As connective proximity and structural role proximity jointly drive the evolution of temporal graphs, hence, the overall loss function is summarized below:

$$\mathcal{L} = \mathcal{L}_c + \lambda \mathcal{L}_r, \tag{10}$$

where λ is the hyper-parameter to balance the connective proximity and structural role proximity.

3.4.3 Implementation. We will further enhance the efficiency of model training through the following two aspects.

Sliding window: In practical application scenarios, particularly in situations with an extensive number of snapshots, it is feasible to employ a suitable sliding window instead of considering the entire historical information from beginning to end. Therefore, Eq. (8) and Eq. (9) can be respectively rewritten as $\mathcal{L}_c|_{[0,T]} \rightarrow \mathcal{L}_c|_{[T-\Delta T,T]}$ and $\mathcal{L}_r|_{[0,T]} \rightarrow \mathcal{L}_r|_{[T-\Delta T,T]}$, where ΔT is the sliding window. This approach not only contributes to further accelerating the training process but also ensures the model maintains a consistently outstanding performance. Specific experimental evidence supporting this proposition is presented in Tables 7 and 8 in Appendix B.1.

Parallel random walk: In the process of determining positive and negative samples for edges during random walk, high parallelization can be achieved by simultaneously running multiple threads to generate a large number of random walks. Therefore, our method can be highly efficient in practical scenarios.

3.4.4 Complexity Analysis. The analysis of the RTGCN's time and space complexities is as follows.

Time complexity: We analyze the time complexity of the proposed RTGCN in each timestamp. In Role-based GNN module, according to [27], the complexity is $O(|\varepsilon_{cu}| + |\varepsilon_{cr}| + |E|)$, and in Role-based GRU module, the complexity is $O((|\varepsilon_{cu}|+|\varepsilon_{cr}|+|E|)Nd')$, where N, ε_{cu} and ε_{cr} are the number of nodes and role sets at current snapshot and historical snapshots, E is the set of edges in the original graph, d and d' represent input and output dimensions, respectively. Table 6 in Appendix A illustrates a comparative analysis of the temporal complexities of the models. Overall, RTGCN can be applied to large-scale graph settings.

Space complexity: The space complexity of our proposed RTGCN mainly depends on the number of snapshots of temporal graphs, namely, T. More specifically, as RTGCN applies modified GRU to update GCN parameters on each snapshot, the number of such parameters in our model does not increase with the increasing number of snapshots, that is, $O(dd')$. Moreover, since the current snapshot hypergraph incidence matrix $\boldsymbol{H}_{cu} \in \mathbb{R}^{N\times|\varepsilon_{cu}|}$ and historical snapshots hypergraph incidence matrix $\boldsymbol{H}_{cr} \in \mathbb{R}^{N\times|\varepsilon_{cr}|}$ are shared across all snapshots, the space complexity of RTGCN can be summarized as $O(dd' + N \times (|\varepsilon_{cu}| + |\varepsilon_{cr}|))$, where $|\varepsilon_{cu}|$ is the number structural role set in current snapshot, and $|\varepsilon_{cr}|$ is the number structural role set in historical snapshots. Please note that $|\varepsilon_{cu}| \ll N$ and $|\varepsilon_{cr}| \ll N$.

4 EXPERIMENTS

Our investigation focuses on addressing the following research questions: **RQ1**: How does the performance of RTGCN compare with that of existing methods? **RQ2**: What is the individual contribution of the various components within RTGCN to its overall performance? **RQ3**: How does RTGCN respond to alterations in hyperparameter settings? **RQ4**: What are the implications of RTGCN's performance in the context of visualization and actual runtime efficiency?

4.1 Experimental Setup

4.1.1 Datasets. We conduct extensive experiments on nine public and popular real-world datasets to assess the quality of RTGCN in dynamic link prediction, dynamic new link prediction, node classification, and structural role classification tasks. As shown in Table 1, these datasets include UCI, Epinions, Alibaba, ML-10M, Facebook, DBLP-3, DBLP-5, America-Air, and Europe-Air. More details about the model implementation and experiment settings are presented in Appendix A.1 and A.2, respectively.

Table 1: Statistics of datasets.

Dataset	#Nodes	#Edges	#Labels	#Features	#Snapshots
UCI	1,809	16,822	/	/	12
Epinions	9,398	231,537	/	44	9
Alibaba	5,640	53,049	/	/	11
ML-10M	20,537	43,760	/	/	9
Facebook	60,730	607,487	/	/	27
DBLP-3	4,257	23,540	3	100	10
DBLP-5	6,606	42,815	5	100	10
America-Air	1,190	13,599	4	/	10
Europe-Air	399	5,995	4	/	10

4.1.2 Baselines. We employ the following methods as the baseline: GCRN [23], DynAERNN [5], DySAT [22], EvolveGCN [17], CTGCN [12], GAEN [24], HTGN [29], MTSN [13], DGCN [3], ROLAND [32], and SpikeNet [10].

4.1.3 Evaluation Tasks and Metrics. We obtain node representations from RTGCN which can be applied to various downstream tasks. In temporal graph representation learning, link prediction is widely used for evaluation. Similar to HTGN, we evaluate our proposed models on dynamic link prediction and dynamic new link prediction tasks. More specifically, given partially observed snapshots of a temporal graph $\mathcal{G} = \{\mathcal{G}_1, ..., \mathcal{G}_t\}$, dynamic link prediction task is defined to predict the link in the next snapshots $\mathcal{G}_{t+1}$ and dynamic new link prediction task is to predict new links in $\mathcal{G}_{t+1}$ that are not in $\mathcal{G}_t$. Similar to GAEN and TRRN [28], *the node classification tasks share a similar objective function with link prediction, the specific objective functions can be found in Eq. (10)*, and we can conduct link prediction and node classification tasks at the same time. Following MTSN and GAEN, we use the area under the ROC curve (AUC) and the F1 score as evaluation metrics in the link prediction task, and the AUC and accuracy (ACC) are utilized to evaluate the node classification performance.

4.2 RQ1: Link Prediction and Node Classification

4.2.1 Link Prediction. We report link prediction results on seven datasets where the best results are indicated in bold and the second-best results are underlined, as illustrated in Tables 2 and 3.

Dynamic Link Prediction: As illustrated in Table 2, it can be observed that the proposed RTGCN significantly outperforms other compared methods considering both AUC and F1 on seven datasets, and has higher stability. For instance, RTGCN achieves an average gain of 3.28% in F1 compared to the best baseline. This demonstrates that the RTGCN can capture structure role information in temporal graphs, which is conducive to obtaining expressive node representations to predict link formation and deletion.

Table 2: Performance evaluation for dynamic link prediction tasks. The results of these baselines methods (dynAE, dynAERNN, DySAT, TSN, and MTSN) are obtained from MTSN. OOT: Out Of Time (72 hours). Please note that these methods of not showing standard deviation are due to the original paper does not provide this information.

Methods	UCI		Alibaba		Epinions		ML-10M		DBLP3		DBLP5		Facebook	
	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1
GCRN	0.804±0.005	0.728±0.002	0.711±0.004	0.647±0.004	0.924±0.002	0.871±0.001	0.797±0.006	0.714±0.005	0.799±0.011	0.726±0.008	0.832±0.005	0.755±0.006	0.687±0.007	0.635±0.006
DySAT	0.728	0.668	0.815	0.770	0.910	0.847	0.870	0.799	0.687±0.009	0.649±0.010	0.701±0.021	0.682±0.013	0.655±0.013	0.611±0.011
dynAE	0.582	0.615	0.645	0.615	0.556	0.663	OOT	OOT	0.742±0.004	0.713±0.005	0.796±0.006	0.732±0.007	0.785±0.018	0.723±0.023
dynAERNN	0.536	0.550	0.584	0.564	0.532	0.644	OOT	OOT	0.638±0.016	0.624±0.014	0.624±0.024	0.671±0.037	0.605±0.014	0.523±0.018
EvolveGCN	0.659±0.021	0.642±0.019	0.876±0.007	0.827±0.009	0.647±0.011	0.637±0.013	0.777±0.011	0.740±0.009	0.825±0.009	0.752±0.013	0.775±0.009	0.714±0.007	0.796±0.008	0.718±0.007
CTGCN-C	0.864±0.008	0.768±0.011	0.821±0.002	0.730±0.001	0.968±0.003	0.905±0.004	0.897±0.007	0.857±0.006	0.777±0.003	0.716±0.006	0.784±0.006	0.712±0.003	0.846±0.006	0.784±0.004
CTGCN-S	0.862±0.013	0.778±0.009	0.819±0.004	0.745±0.003	0.977±0.000	0.922±0.001	0.865±0.005	0.758±0.008	0.723±0.007	0.701±0.005	0.780±0.003	0.733±0.003	0.764±0.005	0.696±0.006
GAEN	0.760±0.006	0.727±0.005	0.806±0.006	0.733±0.004	0.747±0.008	0.713±0.010	0.795±0.014	0.712±0.008	0.783±0.011	0.721±0.012	0.846±0.013	0.771±0.009	0.741±0.011	0.679±0.009
HTGN	0.829±0.002	0.772±0.002	0.884±0.007	0.832±0.008	0.926±0.002	0.867±0.003	0.864±0.012	0.784±0.012	0.751±0.019	0.687±0.010	0.851±0.012	0.777±0.011	0.817±0.013	0.725±0.009
TSN	0.847	0.766	0.871	0.832	0.919	0.862	0.903	0.821	0.873±0.017	0.773±0.012	0.863±0.028	0.764±0.011	0.735±0.020	0.701±0.011
MTSN	0.859	0.777	0.886	0.842	0.931	0.879	0.916	0.862	0.881±0.012	0.782±0.009	0.874±0.016	0.769±0.008	0.769±0.013	0.716±0.009
DGCN	0.613±0.002	0.622±0.005	0.810±0.005	0.739±0.004	0.734±0.008	0.660±0.006	0.724±0.006	0.715±0.009	0.838±0.007	0.791±0.005	0.804±0.009	0.742±0.007	0.677±0.007	0.651±0.008
ROLAND-s	0.834±0.019	0.766±0.016	0.838±0.020	0.786±0.023	0.926±0.025	0.860±0.021	0.824±0.012	0.734±0.014	0.896±0.021	0.853±0.011	0.890±0.011	0.837±0.015	0.785±0.014	0.729±0.013
ROLAND-l	0.854±0.012	0.774±0.010	0.888±0.009	0.824±0.006	0.938±0.009	0.881±0.004	0.836±0.007	0.755±0.010	0.904±0.015	0.864±0.012	0.895±0.005	0.843±0.006	0.793±0.009	0.724±0.008
SpikeNet	0.804±0.017	0.721±0.013	0.839±0.005	0.768±0.003	0.862±0.011	0.803±0.014	0.813±0.006	0.737±0.008	0.826±0.011	0.781±0.009	0.816±0.013	0.769±0.011	0.709±0.010	0.697±0.009
RTGCN (ours)	**0.877±0.002**	**0.796±0.003**	**0.902±0.002**	**0.864±0.001**	**0.989±0.001**	**0.960±0.001**	**0.928±0.005**	**0.889±0.006**	**0.937±0.001**	**0.901±0.001**	**0.946±0.003**	**0.898±0.001**	**0.857±0.004**	**0.793±0.003**

Table 3: Performance evaluation for dynamic new link prediction tasks.

Methods	UCI		Alibaba		Epinions		ML-10M		DBLP3		DBLP5		Facebook	
	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1
GCRN	0.584±0.005	0.563±0.006	0.592±0.006	0.565±0.004	0.763±0.008	0.694±0.006	0.716±0.009	0.611±0.011	0.765±0.005	0.698±0.004	0.791±0.007	0.719±0.005	0.605±0.007	0.576±0.006
DySAT	0.704±0.004	0.651±0.002	0.809±0.007	0.761±0.006	0.896±0.009	0.829±0.007	0.798±0.004	0.707±0.005	0.654±0.004	0.625±0.005	0.651±0.006	0.641±0.005	0.642±0.005	0.603±0.008
dynAE	0.539±0.006	0.524±0.008	0.632±0.023	0.601±0.011	0.531±0.006	0.634±0.003	0.731±0.013	0.633±0.011	0.723±0.011	0.657±0.004	0.768±0.013	0.670±0.005	0.672±0.012	0.617±0.007
dynAERNN	0.524±0.009	0.517±0.005	0.535±0.009	0.543±0.019	0.519±0.002	0.627±0.004	0.740±0.014	0.589±0.010	0.570±0.013	0.608±0.007	0.601±0.012	0.661±0.002	0.554±0.013	0.576±0.006
EvolveGCN	0.597±0.002	0.610±0.004	0.690±0.012	0.666±0.010	0.550±0.004	0.610±0.004	0.617±0.013	0.595±0.008	0.741±0.014	0.700±0.012	0.759±0.005	0.702±0.012	0.774±0.010	0.706±0.011
CTGCN-C	0.668±0.004	0.571±0.004	0.766±0.006	0.692±0.004	0.836±0.004	0.722±0.005	0.813±0.009	0.702±0.010	0.745±0.006	0.685±0.005	0.760±0.006	0.691±0.005	0.760±0.007	0.709±0.006
CTGCN-S	0.626±0.003	0.600±0.002	0.777±0.003	0.711±0.009	0.831±0.005	0.729±0.008	0.758±0.007	0.652±0.009	0.715±0.002	0.654±0.003	0.763±0.003	0.711±0.002	0.654±0.004	0.610±0.005
GAEN	0.548±0.003	0.529±0.002	0.737±0.004	0.704±0.003	0.611±0.005	0.601±0.004	0.741±0.009	0.639±0.006	0.768±0.006	0.704±0.004	0.819±0.004	0.745±0.005	0.712±0.006	0.633±0.005
HTGN	0.815±0.008	0.754±0.013	0.835±0.010	0.797±0.008	0.880±0.001	0.823±0.004	0.850±0.012	0.766±0.005	0.700±0.018	0.649±0.013	0.800±0.015	0.738±0.018	0.804±0.012	0.714±0.011
TSN	0.702±0.011	0.651±0.010	0.778±0.008	0.734±0.009	0.812±0.003	0.765±0.005	0.808±0.005	0.715±0.006	0.690±0.004	0.682±0.007	0.753±0.004	0.716±0.002	0.729±0.005	0.693±0.004
MTSN	0.748±0.008	0.706±0.006	0.801±0.006	0.776±0.005	0.865±0.004	0.812±0.004	0.764±0.004	0.732±0.003	0.824±0.004	0.743±0.005	0.812±0.005	0.738±0.006	0.756±0.006	0.708±0.005
DGCN	0.584±0.002	0.612±0.002	0.803±0.003	0.726±0.004	0.707±0.004	0.619±0.003	0.652±0.004	0.661±0.003	0.832±0.004	0.773±0.004	0.798±0.002	0.727±0.003	0.643±0.004	0.632±0.004
ROLAND-s	0.812±0.005	0.754±0.005	0.814±0.011	0.766±0.012	0.910±0.009	0.846±0.005	0.801±0.006	0.716±0.009	0.859±0.007	0.836±0.009	0.879±0.009	0.822±0.010	0.752±0.007	0.703±0.008
ROLAND-l	0.839±0.008	0.762±0.003	0.843±0.005	0.812±0.004	0.917±0.003	0.850±0.005	0.812±0.006	0.737±0.006	0.871±0.004	0.842±0.003	0.885±0.003	0.822±0.006	0.769±0.004	0.713±0.005
SpikeNet	0.745±0.011	0.709±0.013	0.822±0.002	0.753±0.003	0.835±0.004	0.779±0.003	0.780±0.006	0.713±0.009	0.777±0.010	0.728±0.008	0.797±0.003	0.745±0.008	0.689±0.006	0.673±0.009
RTGCN (ours)	**0.863±0.004**	**0.785±0.003**	**0.891±0.002**	**0.836±0.003**	**0.948±0.002**	**0.897±0.004**	**0.887±0.006**	**0.825±0.005**	**0.924±0.003**	**0.885±0.005**	**0.928±0.001**	**0.884±0.002**	**0.824±0.005**	**0.749±0.006**

Table 4: Performance evaluation for node classification tasks.

Methods	DBLP-3		DBLP-5	
	ACC	AUC	ACC	AUC
GCRN	0.562±0.008	0.738±0.005	0.626±0.005	0.836±0.003
DySAT	0.549±0.016	0.729±0.018	0.520±0.008	0.765±0.016
dynAE	0.389±0.005	0.612±0.009	0.347±0.003	0.623±0.005
dynAERNN	0.321±0.024	0.482±0.017	0.272±0.012	0.501±0.015
EvolveGCN	0.547±0.011	0.765±0.006	0.638±0.011	0.847±0.012
CTGCN-C	0.630±0.003	0.781±0.007	0.665±0.010	0.857±0.005
CTGCN-S	0.591±0.011	0.756±0.004	0.650±0.005	0.854±0.002
GAEN	0.557±0.007	0.693±0.008	0.524±0.009	0.776±0.010
HTGN	0.556±0.003	0.712±0.004	0.538±0.005	0.652±0.009
TSN	0.444±0.006	0.519±0.007	0.276±0.009	0.521±0.019
MTSN	0.604±0.004	0.669±0.005	0.501±0.007	0.754±0.010
DGCN	0.613±0.004	0.754±0.005	0.668±0.006	0.859±0.005
ROLAND-s	0.575±0.008	0.745±0.008	0.579±0.010	0.703±0.011
ROLAND-l	0.625±0.009	0.797±0.007	0.634±0.003	0.768±0.005
SpikeNet	0.519±0.007	0.656±0.001	0.548±0.011	0.703±0.007
RTGCN (ours)	**0.676±0.004**	**0.835±0.003**	**0.690±0.003**	**0.875±0.002**

Table 5: Structural role classification tasks.

Methods	America-Air		Europe-Air	
	ACC	AUC	ACC	AUC
GCRN	0.553±0.007	0.701±0.004	0.455±0.009	0.635±0.009
DySAT	0.374±0.016	0.580±0.008	0.301±0.009	0.539±0.010
dynAE	0.563±0.012	0.702±0.011	0.505±0.011	0.645±0.009
dynAERNN	0.522±0.010	0.682±0.006	0.511±0.014	0.663±0.012
EvolveGCN	0.483±0.011	0.646±0.005	0.380±0.019	0.593±0.012
CTGCN-C	0.578±0.017	0.716±0.012	0.502±0.023	0.666±0.020
CTGCN-S	0.571±0.017	0.712±0.011	0.528±0.011	0.687±0.010
GAEN	0.345±0.008	0.565±0.003	0.400±0.024	0.601±0.017
HTGN	0.476±0.007	0.651±0.004	0.350±0.012	0.572±0.009
TSN	0.479±0.011	0.649±0.008	0.507±0.015	0.678±0.010
MTSN	0.562±0.007	0.706±0.006	0.519±0.008	0.683±0.006
DGCN	0.562±0.005	0.708±0.005	0.481±0.012	0.650±0.005
ROLAND-s	0.528±0.007	0.677±0.005	0.520±0.016	0.685±0.009
ROLAND-l	0.547±0.005	0.696±0.006	0.526±0.006	0.687±0.003
SpikeNet	0.517±0.004	0.674±0.004	0.403±0.016	0.608±0.013
RTGCN (ours)	**0.605±0.006**	**0.737±0.005**	**0.544±0.005**	**0.698±0.003**

Dynamic New Link Prediction: New link prediction aims to predict the appearance of new links, which is more challenging. From Table 3, we observe analogous observations to the dynamic link prediction task, demonstrating the superiority of the proposed RTGCN. Specifically, all baseline methods experience performance drops to varying degrees compared to the corresponding dynamic link prediction task, while our RTGCN model produces more consistent results. For instance, the performance of the baselines degrades dramatically on Epinions (e.g., the second-best, CTGCN-S drops from 0.977 to 0.831), but our RTGCN only declines about 0.041. In addition, compared with the dynamic link prediction task, the performance gain of the dynamic new link prediction task is greater. For instance, RTGCN obtains a 7.7% improvement in terms of F1 compared to the best baseline on ML-10M. Our model shows strong inductive ability via utilizing global structural role information.

4.2.2 Node Classification. To further evaluate the node representation quality, we consider node classification tasks. Note that link prediction and node classification tasks share the objective function.
Node Classification: Table 4 reports the node classification performance on the two labeled graphs, where the node classification performance is much lower compared with the respective link prediction results in Table 2 and Table 3. This can be attributed to the same training mechanism employed by both compared methods and our proposed model. That is, nodes are compelled to have similar embedding with their linked neighborhoods via connective proximity (e.g., Eq. (8)). Therefore, the training scheme would be more inclined to link prediction task, which is consistent with the results of other research [24, 28]. We can observe that RTGCN outperforms all baseline methods, which further asserts the superiority of RTGCN.
Structural Role Classification: To further evaluate the structural role similarity preserving ability, we test the dynamic structural role classification task in the European air-traffic network and American air-traffic network. In this task, the learned node representations are utilized to predict the structure role-related labels of nodes in each temporal graph. As shown in Table 5, we can observe that our proposed RTGCN outperforms all compared baselines on both datasets, which indicates that RTGCN can preserve global structural similarity between nodes in temporal graphs. Furthermore, CTGCN-S and CTGCN-C also attain the second-best performance in terms of structural role classification, which is consistent with the analysis results of previous studies [12]. This demonstrates that modeling global structural similarity and employing structural role proximity constraints are effective ways to distinguish nodes.

4.3 RQ2: Ablation Study

To evaluate the effectiveness of different components in RTGCN, we conduct the ablation study with several variants which are introduced as follows: 1) **W/O RGR**: RTGCN replaces the role-based GRU module by only taking the hidden state of the last snapshot and adjacency matrix as input. 2) **W/O RCU**: RTGCN does not consider the current snapshot hypergraph in the role-based GNN module. 3) **W/O RCR**: RTGCN does not consider the historical snapshots hypergraph in the role-based GNN module. 4) **W/O RRP**: without the role proximity, ie., RTGCN is trained only using the connective proximity. 5) **W/O RAM**: RTGCN does not use adjacency matrix in role-based GNN module and role-based GRU module. As shown in Figure 3 , we have the following observations: 1) RTGCN achieves the best performance when it is equipped with all the components, and removing any component would cause worse results. 2) Interestingly, when we only use the role-based incidence matrix for model training, our model still achieves better performance than all baselines on DBLP-3. In addition, RTGCN also obtains a comparable performance on Alibaba. While on America-Air, the performance has declined, and the potential reason is that local structural changes dominate the evolution of the graph.

4.4 RQ3: Parameter Sensitivity

In this section, we further investigate the effect of model parameters on dynamic link prediction, node classification, and structural role classification tasks. Due to the space limit, we only present

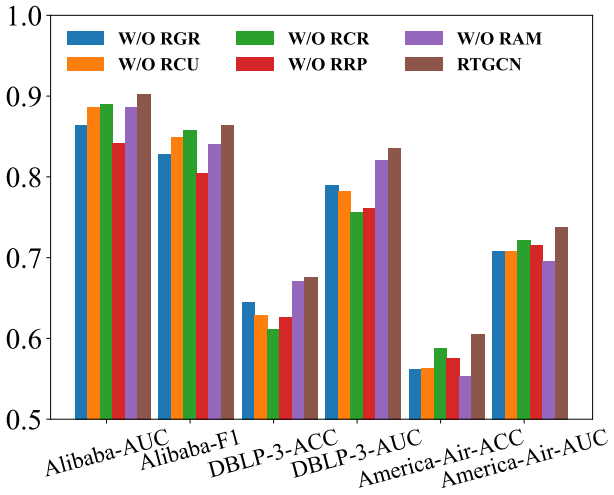


Figure 3: Performance of evaluation for ablation study.

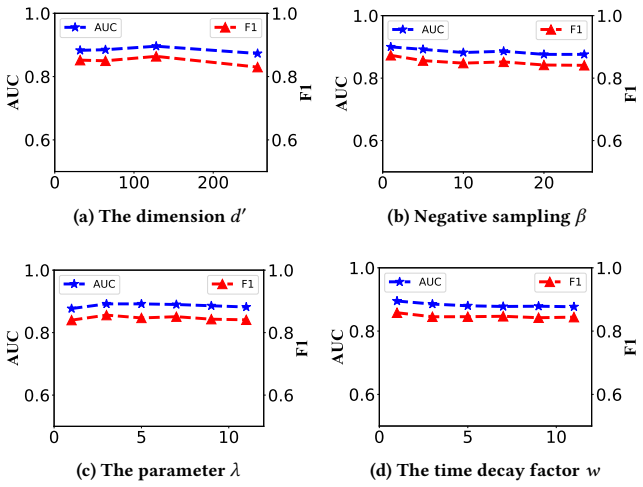
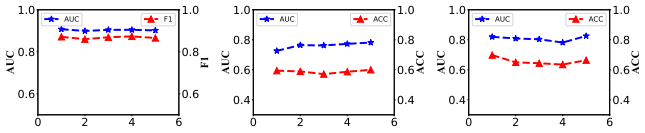


Figure 4: Performance of dynamic link prediction with different parameter settings on the Alibaba.

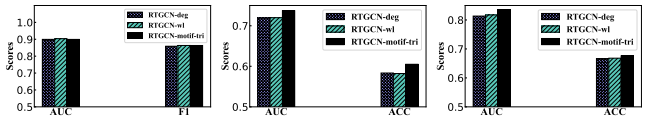
partial results of the parameter study in Figures 4 and 5. More studies regarding model parameters are in Appendix B.2. In general, RTGCN is insensitive to different parameter settings except for time interval Δt. This is expected because an excessive amount of historical information may introduce irrelevant noise, thereby diluting the significance of the most important historical snapshots.

To further evaluate the generalization ability of the proposed model, we use different structural role discovery methods on link prediction, node classification, and structural role classification tasks. As illustrated in Figure 6, RTGCN outperforms all baseline models across different tasks utilizing various role discovery methods, further demonstrating that global structure role information and role proximity can assist in producing more expressive node presentations for various downstream tasks.



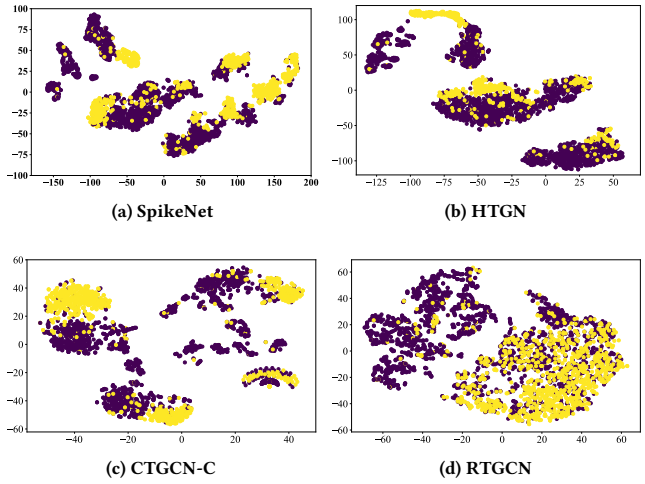
(a) Alibaba (b) America-Air (c) DBLP-3

Figure 5: The impact of Δt on RTGCN performance.



(a) Alibaba (b) America-Air (c) DBLP-3

Figure 6: The impact of different structural role assignment methods on the performance of RTGCN.



(a) SpikeNet (b) HTGN

(c) CTGCN-C (d) RTGCN

Figure 7: 2-D t-SNE projections of finanl node embedding on DBLP-3 in snapshot 6. We randomly select two types of nodes from three types of nodes in the graph, with different colors referring to different classes.

4.5 RQ4: Visualization and Efficiency Comparison

In the visualization task, we utilize t-SNE [25] to project node embeddings of temporal graphs into a 2D space, while selecting three SOTA baseline for comparison. From the results shown in Figure 7, we observe that RTGCN can more clearly separate the nodes belonging to different classes, which further demonstrates that global structure role information is beneficial to distinguish nodes.

Figure 8 shows the training time of each epoch for different methods on DBLP-3 for the node classification task and Alibaba for the dynamic link prediction task. Please note that in our proposed model, namely RTGCN, the runtime of the structural role discovery process is also included. Based on the experimental results, it can be observed that our proposed model achieves an excellent balance between model performance and efficiency. The RTGCN demonstrates comparable efficiency to the CTGCN, ROLAND, and SpikeNet. This achievement can be attributed to the following factors: (1) we leverage structural role-based GRU module to update the model parameters instead of training from scratch at each snapshot. (2) compared with GAEN, MTSN, and CTGCN, our model benefits from a lightweight model architecture, with fewer computational costs and parameters to optimize.

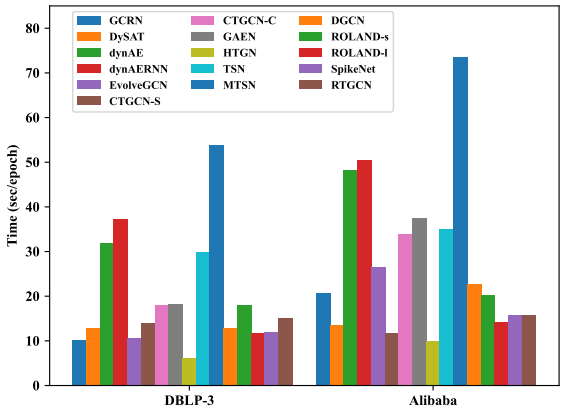


Figure 8: Running time of different methods.

5 CONCLUSION

In this work, we propose a novel structural role-based node representation learning model for the temporal graph. To our best knowledge, RTGCN is the first research effort to leverage the global structural role information on temporal graphs. More specifically, we first construct the structural role-based hypergraph. Then, we propose two novel modules: the structural role-based GNN module and the role-based GRU module, which respectively model global structural role information and capture graph evolving patterns. All modules are proposed to impel the success of RTGCN. To evaluate our method, we conduct various tasks on multiple real-world temporal graphs. The empirical experiments demonstrate our approach outperforms the SOTA temporal graph representation learning baselines by a large margin. For future work, we intend to generalize our method to explore large-scale continuous temporal graphs. Besides, applying large language models for reasoning tasks on temporal graphs is also an interesting direction.

ACKNOWLEDGMENTS

The research is supported by the Key Technologies Research and Development Program under Grant No. 2020YFC0832702, and National Natural Science Foundation of China under Grant Nos. 71910107002, 62376227, 61906159, 62302400, 62176014, 62201475 and Sichuan Science and Technology Program under Grant No. 2023NSFSC0032, 2023NSFSC0114, and Guanghua Talent Project of Southwestern University of Finance and Economics. Carl Yang was not supported by any funds from China.

REFERENCES

[1] Nesreen K Ahmed, Ryan A Rossi, John Boaz Lee, Theodore L Willke, Rong Zhou, Xiangnan Kong, and Hoda Eldardiry. 2020. Role-based graph embeddings. *TKDE* 34, 5 (2020), 2401–2415. https://doi.org/IEEE

[2] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. 2019. Hypergraph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*. 3558–3565.

[3] Chao Gao, Junyou Zhu, Fan Zhang, Zhen Wang, and Xuelong Li. 2022. A novel representation learning for dynamic graphs based on graph convolutional networks. *IEEE Transactions on Cybernetics* 53, 6 (2022), 3599–3612.

[4] Wulfram Gerstner, Werner M Kistler, Richard Naud, and Liam Paninski. 2014. *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press.

[5] Palash Goyal, Sujit Rokka Chhetri, and Arquimedes Canedo. 2020. dyngraph2vec: Capturing network dynamics using dynamic graph representation learning. *Knowledge-Based Systems* 187 (2020), 104816.

[6] Pengfei Jiao, Xuan Guo, Ting Pan, Wang Zhang, Yulong Pei, and Lin Pan. 2021. A survey on role-oriented network embedding. *IEEE Transactions on Big Data* 8, 4 (2021), 933–952.

[7] Thomas N Kipf and Max Welling. 2016. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

[8] Fuxian Li, Jie Feng, Huan Yan, Guangyin Jin, Fan Yang, Funing Sun, Depeng Jin, and Yong Li. 2023. Dynamic graph convolutional recurrent network for traffic prediction: Benchmark and solution. *ACM Transactions on Knowledge Discovery from Data* 17, 1 (2023), 1–21.

[9] Jundong Li, Harsh Dani, Xia Hu, Jiliang Tang, Yi Chang, and Huan Liu. 2017. Attributed network embedding for learning in a dynamic environment. In *CIKM*. 387–396.

[10] Jintang Li, Zhouxin Yu, Zulun Zhu, Liang Chen, Qi Yu, Zibin Zheng, Sheng Tian, Ruofan Wu, and Changhua Meng. 2023. Scaling Up Dynamic Graph Representation Learning via Spiking Neural Networks. In *AAAI*. 8588–8596.

[11] Jun Liu, Gang Wang, Ping Hu, Ling-Yu Duan, and Alex C Kot. 2017. Global context-aware attention lstm networks for 3d action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1647–1656.

[12] Jingxin Liu, Chang Xu, Chang Yin, Weiqiang Wu, and You Song. 2020. K-core based temporal graph convolutional network for dynamic graphs. *IEEE Transactions on Knowledge and Data Engineering* 34, 8 (2020), 3841–3853.

[13] Zhijun Liu, Chao Huang, Yanwei Yu, and Junyu Dong. 2021. Motif-preserving dynamic attributed network embedding. In *Proceedings of the Web Conference*. 1629–1638.

[14] Xuewei Ma, Geng Qin, Zhiyang Qiu, Mingxin Zheng, and Zhe Wang. 2019. RiWalk: Fast structural node embedding via role identification. In *ICDM*. 478–487.

[15] Yao Ma, Ziyi Guo, Zhaocun Ren, Jiliang Tang, and Dawei Yin. 2020. Streaming graph neural networks. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 719–728.

[16] Giannis Nikolentzos and Michalis Vazirgiannis. 2021. Learning structural node representations using graph kernels. *IEEE transactions on knowledge and data engineering* 33, 5 (2021), 2045–2056.

[17] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao Schardl, and Charles Leiserson. 2020. Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In *AAAI*. 5363–5370.

[18] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. 2019. Geom-GCN: Geometric Graph Convolutional Networks. In *ICLR*.

[19] Yulong Pei, Xin Du, Jianpeng Zhang, George Fletcher, and Mykola Pechenizkiy. 2020. struc2gauss: Structural role preserving network embedding via Gaussian embedding. *Data Mining and Knowledge Discovery* 34 (2020), 1072–1103.

[20] Ryan A Rossi, Nesreen K Ahmed, Eunyee Koh, Sungchul Kim, Anup Rao, and Yasin Abbasi-Yadkori. 2020. A structural graph representation learning framework. In *Proceedings of the 13th international conference on web search and data mining*. 483–491.

[21] Ryan A Rossi, Di Jin, Sungchul Kim, Nesreen K Ahmed, Danai Koutra, and John Boaz Lee. 2020. On proximity and structural role-based embeddings in networks: Misconceptions, techniques, and applications. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 14, 5 (2020), 1–37.

[22] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, and Hao Yang. 2020. Dysat: Deep neural representation learning on dynamic graphs via self-attention networks. In *Proceedings of the 13th international conference on web search and data mining*. 519–527.

[23] Youngjoo Seo, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. 2018. Structured sequence modeling with graph convolutional recurrent networks. In *International conference on neural information processing*. 362–373.

[24] Min Shi, Yu Huang, Xingquan Zhu, Yufei Tang, Yuan Zhuang, and Jianxun Liu. 2021. GAEN: Graph Attention Evolving Networks.. In *IJCAI*. 1541–1547.

[25] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008), 2579–2605.

[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 6000–6010.

[27] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2021. A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems* 32, 1 (2021), 4–24.

[28] Dongkuan Xu, Junjie Liang, Wei Cheng, Hua Wei, Haifeng Chen, and Xiang Zhang. 2021. Transformer-style relational reasoning with dynamic memory updating for temporal network modeling. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 4546–4554.

[29] Menglin Yang, Min Zhou, Marcus Kalander, Zengfeng Huang, and Irwin King. 2021. Discrete-time temporal network embedding via implicit hierarchical learning in hyperbolic space. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 1975–1985.

[30] Menglin Yang, Min Zhou, Hui Xiong, and Irwin King. 2022. Hyperbolic Temporal Network Embedding. *IEEE Transactions on Knowledge and Data Engineering* 35, 11 (2022), 11489–11502.

[31] Qiang Yang, Changsheng Ma, Qiannan Zhang, Xin Gao, Chuxu Zhang, and Xiangliang Zhang. 2023. Interpretable Research Interest Shift Detection with Temporal Heterogeneous Graphs. In *WSDM*. 321–329.

[32] Jiaxuan You, Tianyu Du, and Jure Leskovec. 2022. ROLAND: graph learning framework for dynamic graphs. In *KDD*. 2358–2366.

[33] Hegui Zhang and Gang Kou. 2022. Role-based multiplex network embedding. In *International Conference on Machine Learning*. PMLR, 26265–26280.

[34] Kaike Zhang, Qi Cao, Gaolin Fang, Bingbing Xu, Hongjian Zou, Huawei Shen, and Xueqi Cheng. 2023. DyTed: Disentangled Representation Learning for Discrete-time Dynamic Graph. In *KDD*. 3309–3320.

[35] Wang Zhang, Xuan Guo, Wenjun Wang, Qiang Tian, Lin Pan, and Pengfei Jiao. 2021. Role-based network embedding via structural features reconstruction with degree-regularized constraint. *Knowledge-Based Systems* 218 (2021), 106872.

A REPRODUCIBILITY

In this section, we provide more details of model implementation and experiment setup for the reproducibility of the experiment results. Our implementation code is provided through the link[3].

Algorithm 1: Training the RTGCN model

Input: Temporal graph $\mathcal{G} = \{G_1, ..., G_T\}$, adjacency matrix sequence $\mathcal{A} = \{\boldsymbol{A}_1, ..., \boldsymbol{A}_T\}$ and hypergraph incidence matrix sequence, $\mathcal{H}_{cu} = \{\boldsymbol{H}_{cu,1}, ..., \boldsymbol{H}_{cu,T}\}$ and $\mathcal{H}_{cr} = \{\boldsymbol{H}_{cr,2}, ..., \boldsymbol{H}_{cr,T}\}$.
Output: Embedding of each graph $G_t : \boldsymbol{O}_t \in \mathbb{R}^{n \times d'}$
Initialize the trainable weight matrices and training epoch I.
for $j \in [1, I]$ **do**
  for $t = 1$ **do**
    /* **Structural role-based GNN module** */
    $\boldsymbol{h}_{v,1}^{(l)} \leftarrow$ node embeddings via Eq. (4).
    $\boldsymbol{X}_{cu,1}^{l+1} \leftarrow$ node embeddings via Eq. (5).
    $\boldsymbol{O}_1 \leftarrow$ embedding concatenation via Eq. (7).
  end
  for $t \in [2, T]$ **do**
    /* **Structural role-based GRU module** */
    $\widehat{\boldsymbol{H_t}} = GRU\left(\boldsymbol{A}_t, \boldsymbol{H}_{cu,t}, \boldsymbol{H}_{cr,t}, \widehat{\boldsymbol{H_{t-1}}}\right)$
    /* **Structural role-based GNN module** */
    $\boldsymbol{h}_{v,t}^{(l)} \leftarrow$ compute node embeddings via Eq.(4).
    $\boldsymbol{X}_{cu,t}^{l+1} \leftarrow$ node embeddings via Eq.(5).
    $\boldsymbol{X}_{cr,t}^{l+1} \leftarrow$ node embeddings via Eq. (6).
    $\boldsymbol{O}_t \leftarrow$ embeddings concatenation via Eq. (7).
  end
  Minimized the overall loss in Eq.(10).
end
Return $\{\boldsymbol{O}_t\}$

A.1 Details of Model Implementation

The procedure of training RTGCN is illustrated in Algorithm 1. The detailed implementation contains the following components:

- **Role discovery methods:** The methods for role discovery primarily encompass the following three methods: degree-based, Weisfeiler-Lehman-based, and Motif-count-based [14, 33]. In this study, the role assignment process of the degree-based and motif-count-based methods employs equivalence rules (i.e., two nodes are assigned the same structural role set if their feature vectors are identical), while for the Weisfeiler-Lehman-based method, we assign roles based on the first element of nodes' feature vector. It is important to note that, with the exception of ablation experiments that explore the impact of different role discovery methods on model performance, all experimental results are derived from the Motif-count-based role discovery method.
- **Historical snapshots hypergraph construction:** we set Δt as 1 for simplifying the model calculation. Additionally, we delve deeper into the effects of Δt on model performance within the parameter study section.

[3]https://github.com/trytodoit227/RTGCN-new.

Table 6: Model Complexity Comparison.

Model	Time Complexity
HTGN[29]	$O\left(N^{\prime}\mathcal{W}d' + Nd + Ndd' + d'\,\lvert E\rvert\right)$
CTGCN[12]	$O\left(N\left(d + d'\right)k' + lk\,\lvert E\rvert\right)$
SpikeNet[10]	$O\left(Nd^2S^{\mathbf{k}}\right)$
DGCN[3]	$O\left(\lvert E\rvert c_{avg}\log c_{avg}\right)$
MTSN[13]	$O\left(Ndd' + Ld\,\lvert E\rvert\right)$
RTGCN(our)	$O\left(\lvert E\rvert + \lvert E\rvert\, Nd'\right)$

N is the number of nodes, d and d' represent input and output dimensions, respectively. $|E|$ is the number of edges, S is the neighborhood size, c_{avg} is the average degree of node, l is the layer number of the CGCN layers, k' is a constant independent of N, $\mathbf{k}$ is layer SpikeNet, $\mathcal{W}$ denotes state memory length. ***Please note that the reason why these methods do not present model complexity is that the original paper does not provide this information.***

The comparison of time complexities for different models is illustrated in Table 6. If prediction tasks are necessary for each snapshot, the overall time complexity would be $O\left(T\left(|E| + |E|\, Nd'\right)\right)$. If we adopt a sliding window approach, then the model's parameters are reduced to $O\left(\Delta T\left(|E| + |E|\, Nd'\right)\right)$. It can be observed that the time complexity of our method is proportional to the number of edges, comparable to the time complexities of other baselines, making it suitable for large-scale graph tasks.

A.2 Experiment Settings

Node initial feature: In our experiment, most of the benchmark datasets for temporal graph representation learning are only associated with topology. For UCI, Alibaba, ML-10M, Facebook, America-Air, and Europe-Air, we employ identity matrix as the node feature which is identical to the processing in [30, 32]. Conversely, for DBLP3, DBLP5, and Epinions, their nodes are associated with node features and we directly use them in our work.
Dataset splitting: For the link prediction task, following the same setting as in MTSN [13], we use 20% links in the graph at the next snapshot as the validation set to tune model hyperparameters. Furthermore, we randomly sample another 20% links for training and use the remaining 60% for testing. For node classification task, 70% of the nodes are used for training. Then, an additional 20% nodes and the remaining 10% nodes are allocated for validation and testing, respectively.
Parameter settings: We follow the methodology in [12] and compute edge feature vectors by utilizing the Hadamard operation between embedding vectors of node pairs within labeled edge sets. We set the final embedding dimension of HTGN [29] as 16, and other compared models as 128. To ensure an equitable comparison, the maximum number of training epochs for all models is set at 200. For all random walk-based methods, we set the window size as 6. In addition, the number of negative sampling, walks, and walk lengths are set to 8, 100 and 10, respectively. We utilize 2 GCN layers in the GCRN and EvolveGCN. In CTGCN, we utilize 2 CGCN layers in the CTGCN-C and CTGCN-S and the k-core subgraph number of the CGCN layer is set as the maximum k-core number across all

Table 7: Dynamic link prediction tasks.

Snapshots	UCI		DBLP3		DBLP5	
	AUC	F1	AUC	F1	AUC	F1
$[T-\Delta T, T]$	0.867	0.788	**0.941**	0.892	**0.951**	0.889
$[0, T]$	**0.877**	**0.796**	0.937	**0.901**	0.946	**0.898**

Table 8: Dynamic new link prediction tasks.

Snapshots	UCI		DBLP3		DBLP5	
	AUC	F1	AUC	F1	AUC	F1
$[T-\Delta T, T]$	0.855	**0.789**	**0.928**	0.884	0.916	0.874
$[0, T]$	**0.863**	0.785	0.924	0.885	**0.928**	**0.884**

snapshots. We set β as 5, the look back value as 1, γ_1 as 10^{-6} and γ_2 as 10^{-6} for dynAE and dynAERNN. The default parameter settings of other compared models suggested by the original paper were used. For the compared methods, we use the source code released by the authors for baseline evaluation.

In RTGCN, the final embedding dimension d' is set at 24, and the number of negative sampling β and walk lengths are configured to 10 and 20, respectively. λ is set to 5 for Alibaba, 5 for DBLP-3, and 1 for America-Air. For the dynamic link prediction task, in our prediction layer, we only train a logistic regression classifier to predict whether there exists a link between each node pair in the testing set. As for the node classification task, we train a logistic regression classifier to classify nodes into different categories based on embedding features acquired from previous graphs up to time t. This aligns with the methodology employed in previous work [24]. For model optimization, we employ the Adam optimizer with a learning rate of 0.008 and weight decay of 3×10^4.

Throughout the experimentation, we performed the testing procedure 10 times and reported the average performance as the final model result. Notably, both the baselines and RTGCN models are uniformly trained using early stopping criteria based on validation set performance.

B MORE EXPERIMENTAL RESULTS

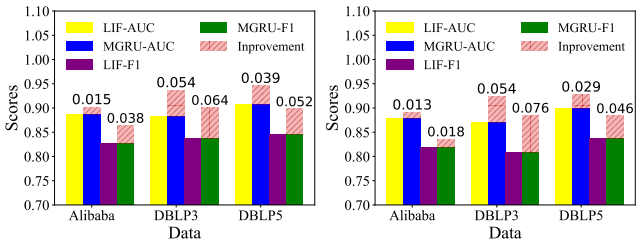
B.1 Ablation Study

It is well known that not all historical information is beneficial for predicting the next snapshot. Simultaneously, to demonstrate the efficiency of our model, in the edge prediction task, we consider the information from the previous ΔT snapshots as historical data and compare it with using all historical snapshots. As shown in Tables 7 and 8, our approach achieves comparable or even superior performance by solely utilizing information from the previous $\Delta T = 1$ snapshots, highlighting the efficiency of our method.

To further validate the effectiveness of our proposed role-based GRU module in modeling the dynamic evolution patterns of graphs, we compared it with the state-of-the-art LIF [10] model while keeping other modules unchanged. The experimental results on different datasets, as shown in Figure 9, demonstrate that our role-based GRU module is simpler and more effective.

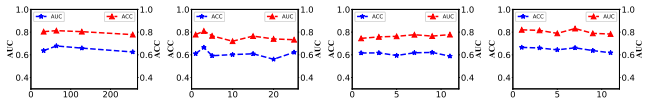
B.2 Parameter Study

Here, we conduct further parameter studies on DBLP-3 and America-Air datasets. Specifically, we explore variations in the negative



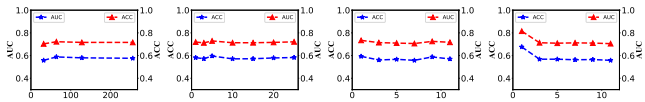
(a) Dynamic link prediction tasks (b) Dynamic new link prediction tasks.

Figure 9: LIF represents the modeling of temporal features in temporal graphs using the LIF model, while MGRU denotes the utilization of the GRU module proposed in this paper, which considers structural roles, for temporal modeling.



(a) d' (b) β (c) λ (d) w

Figure 10: Performance of node classification with different parameter settings on the DBLP-3.



(a) d' (b) β (c) λ (d) w

Figure 11: Performance of structural role classification with different parameter settings on the America-Air.

sampling parameter β from the set $\{1, 3, 5, 10, 15, 20, 25\}$, as well as the representation dimension within the set $\{32, 64, 128, 256\}$. Furthermore, we examine the impact of λ and the time decay factor w across $\{1, 3, 5, 7, 9, 11\}$. In addition, we also explore the impact of historical information from different time intervals Δt on the performance of our model, and the Δt is tuned in $\{1, 2, 3, 4, 5\}$. From the results shown in 10 and 11, we can observe that the performance of RTGCN is generally stable under different parameter settings, except for the time decay factor w and time interval Δt.

To further validate the effectiveness of the proposed model, we use different role discovery algorithms for node role assignment. Here, we employ three role discovery methods (RTGCN-deg: degree-based, RTGCN-wl: Weisfeiler-Lehman-based, and, RTGCN-motif-tri: Motif-count-based) to conduct dynamic link prediction, node classification, and structural role classification tasks, respectively. As shown in Figure 6 in section 4.4, our model consistently surpasses all baseline methods under different role discovery methods. These observations further validate the effectiveness of structural role information and the robustness of our method.