Privacy-Preserved Evolutionary Graph Modeling via Gromov-Wasserstein Autoregression

Yue Xiang[1], **Dixin Luo**[2], **Hongteng Xu**[3, 4*]

[1]School of Statistics, Renmin University of China
[2]School of Computer Science and Technology, Beijing Institute of Technology
[3]Gaoling School of Artificial Intelligence, Renmin University of China
[4]Beijing Key Laboratory of Big Data Management and Analysis Methods
xiangyue@ruc.edu.cn, dixin.luo@bit.edu.cn, hongtengxu@ruc.edu.cn

Abstract

Real-world graphs like social networks are often evolutionary over time, whose observations at different timestamps lead to graph sequences. Modeling such evolutionary graphs is important for many applications, but solving this problem often requires the correspondence between the graphs at different timestamps, which may leak private node information, e.g., the temporal behavior patterns of the nodes. We proposed a Gromov-Wasserstein Autoregressive (GWAR) model to capture the generative mechanisms of evolutionary graphs, which does not require the correspondence information and thus preserves the privacy of the graphs' nodes. This model consists of two autoregressions, predicting the number of nodes and the probabilities of nodes and edges, respectively. The model takes observed graphs as its input and predicts future graphs via solving a joint graph alignment and merging task. This task leads to a fused Gromov-Wasserstein (FGW) barycenter problem, in which we approximate the alignment of the graphs based on a novel inductive fused Gromov-Wasserstein (IFGW) distance. The IFGW distance is parameterized by neural networks and can be learned under mild assumptions, thus, we can infer the FGW barycenters without iterative optimization and predict future graphs efficiently. Experiments show that our GWAR achieves encouraging performance in modeling evolutionary graphs in privacy-preserving scenarios.

Introduction

Graphs are capable of describing relational and structured information hidden in data. In many real-world scenarios, such as social networks (Tantipathananandh, Berger-Wolf, and Kempe 2007) and financial networks (Durante and Dunson 2014), the topological structure of a graph is often evolutionary — the graph size, the attributes of its nodes, and the edges between the nodes may change over time according to the temporal behaviors of nodes. Such evolutionary graphs (also known as dynamic graphs or temporal graphs) are recorded as sequences of observed graphs.

The key problem of evolutionary graph modeling is predicting future graphs given a sequence of observed graphs. Currently, most existing modeling methods, $e.g.$, JODIE (Kumar, Zhang, and Leskovec 2019), VGRNN (Hajiramezanali et al. 2019) and DynAERNN (Goyal, Chhetri, and Canedo 2020), formulate the problem as a link prediction task, capturing the topological dynamics of evolutionary graphs by predicting the connections between arbitrary two nodes based on their historical behaviors. Although these methods achieve encouraging performance on evolutionary graph modeling, they often suffer from a risk of private information leakage. In particular, the above methods require the correspondence across the observed graphs, making the temporal behaviors of each node become accessible. It has been demonstrated that the privacy of nodes ($e.g.$, the profiles of users in a network) can be inferred from their temporal behaviors (Cheng, Caverlee, and Lee 2010; Paul and Dredze 2011; Wang et al. 2013; Luo et al. 2014).

To accomplish privacy protection, we need to model evolutionary graph from *unaligned* graph sequences — the correspondence between arbitrary two graphs' nodes is unknown. Such a privacy-preserved evolutionary graph modeling task is meaningful in practice. Take social network modeling as an example. With the registration of new accounts and the expiration of old ones, the topological change of a social network comprises both nodes and edges. At different timestamps, not only the edges between the accounts but also the accounts themselves may change. Modeling and predicting the evolution of such social networks are significant for both the commercial operation of service providers and the supervision of third parties, and how to protect user privacy throughout the modeling phase is one of the primary challenges associated with the above activities.

In this work, we propose a Gromov-Wasserstein autoregressive (GWAR) model, treating unaligned evolutionary graph modeling as a statistically-interpretable graph generation task. As illustrated in Figure 1, our GWAR model consists of two autoregressions oriented to the number of nodes ($a.k.a$, graph size) and graph structure, respectively. Given a sequence of observed graphs, the size-oriented autoregression predicts the number of nodes for the future graph. After determining the graph size, the structure-oriented autoregression predicts the probabilities of nodes and edges, in which the historical graphs are aligned and merged together. In particular, the joint alignment of the historical graphs is achieved as a fused Gromov-Wasserstein (FGW) barycenter problem (Vayer et al. 2020) under a novel inductive fused

*Corresponding author.

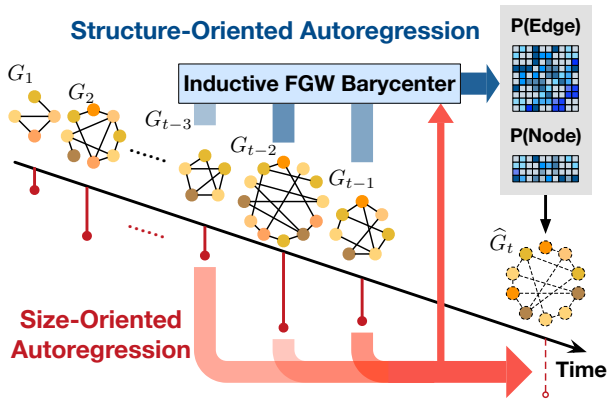


Figure 1: An illustration of the proposed GWAR model.

Gromov-Wasserstein (IFGW) distance. The optimal transport associated with each IFGW distance is parameterized by a graph neural network (GNN) with coupled attention layers, which indicates the correspondence between graphs. After learning the IFGW distance under mild assumptions, we can infer the optimal transport without iterative optimization and align graphs efficiently. The IFGW distance is significant to improve the efficiency of our GWAR model on both learning and inference, which avoids iterative optimization used in conventional FGW distance. As a result, we successfully simplified the learning of the GWAR model from a bi-level optimization problem to two independent single-level problems. Experimental results show that our GWAR model outperforms state-of-the-art methods when modeling unaligned evolutionary graphs.

Related Work

Graph sequence modeling Many graph models have been developed for generative tasks, $e.g.$, probabilistic models (Erdos, Rényi et al. 1960), stochastic block models (Holland, Laskey, and Leinhardt 1983), adversarial network over random walks (NetGAN) (Bojchevski et al. 2018), and variational autoencoders (GraphVAE) (Simonovsky and Komodakis 2018). Among them, some models focus on graph sequences, such as the graph recurrent attention network (GRAN) in (Liao et al. 2019), the TagGen for temporal interaction network (Zhou et al. 2020), and the temporal graph generative adversarial network (TG-GAN) in (Zhang et al. 2021). These models describe the evolving sequences of graphs and predict future graphs based on historical observations. Additionally, link prediction for dynamic graphs is a task highly correlated with evolutionary graph modeling. Representative link prediction methods, $e.g.$, variational graph recurrent neural networks (VGRNN) (Hajiramezanali et al. 2019) and EvolveGCN (Pareja et al. 2020), learn node embeddings and predict edge probabilities based on the embeddings. However, the aforementioned methods assume that the graph size is fixed during the evolutionary process and the correspondence between different graphs is known, which is questionable for privacy protection.

Privacy-preserved graph modeling Extensive user data are collected for services in various domains such as healthcare (Li et al. 2020), banking systems (Wang et al. 2021) and bioinformatics (Li et al. 2021), which leads to powerful graph-based models and a high risk of privacy issue at the same time. Typically, differential privacy (DP) (Dwork et al. 2006) is used to prevent information leaks in machine learning models by adding "just enough" noise during model training. When learning GNNs, the DP strategy is applied either to the graph data directly (Qiu et al. 2022) or to the gradient of model parameters (Igamberdiev and Habernal 2021). Recently, federated learning has also been extended to train GNNs to protect user privacy, $e.g.$, FedGNN (Wu et al. 2021) incorporates the high-order user-item interactions by building the local user-item graphs in a privacy-preserving way. Besides, some other privacy-preserved graph modeling methods encodes sensitive information before training, $e.g.$, the work in (Shen, Leus, and Giannakis 2019) only requires an encrypted version of each node's connectivity and hence promotes node privacy. However, none of the above methods suppress the risk of privacy leakage caused by observing the temporal behaviors of nodes based on the correspondence across observed graphs. Currently, modeling users' behavior sequences to capture their behavior patterns is important to discover potential interests in recommendation and social networks, while the privacy issue in such scenarios is not fully considered yet.

Gromov-Wasserstein graph modeling As an important variant of optimal transport distance, Gromov-Wasserstein (GW) distance (Mémoli 2011; Chowdhury and Mémoli 2019) provides an useful pseudometric for graphs, which has been applied to many problems, $e.g.$, registering shapes (Mémoli 2011), aligning protein networks (Xu et al. 2019), and matching vocabulary sets of different languages (Alvarez-Melis and Jaakkola 2018). Given multiple graphs, the work in (Peyré, Cuturi, and Solomon 2016) proposed a new concept called Gromov-Wasserstein barycenter, achieving the interpolation of multiple graphs. Based on GW barycenters, a multi-graph matching method is proposed in (Xu, Luo, and Carin 2019). More recently, focusing on the graphs with node attributes, the work in (Titouan et al. 2019) proposes fused GW (FGW) distance, which combines the GW distance between graph structures with the Wasserstein distance (Villani 2008) between node attributes. The FGW distance is proven to be useful in graph clustering and classification tasks. In the aspect of computation, most of the existing works calculate GW distance by Sinkhorn iterations(Sinkhorn and Knopp 1967), whose complexity per iteration is $\mathcal{O}(N^3)$ for the graphs with N nodes. The high computational complexity limits the applications of GW distance. These years, many variants of GW distance have been proposed, $e.g.$, the recursive GW distance (Xu, Luo, and Carin 2019), the sliced GW distance (Vayer et al. 2019), and the Bregman ADMM-based algorithm (Xu 2020).

Gromov-Wasserstein Autoregression

Proposed Privacy-Preserved Modeling Strategy

Denote unaligned evolutionary graphs as a graph sequence $\{G_t\}_{t=0}^{T}$, where the correspondence between arbitrary two

graphs is unknown. We are interested in modeling such unaligned evolutionary graphs in an autoregressive (AR) manner, generating each G_t or its *isomorphism* based on its K-step history $\{G_{t-k}\}_{k=1}^{K}$, *i.e.*,

$$G_t \cong f(G_{t-1}, \dots, G_{t-K}). \tag{1}$$

Here, $\cong$ means isomorphism, which means that we can obtain G_t by permuting the nodes of the generated graph. The AR model f generates an isomorphism of G_t rather than the exact G_t because of the requirement of privacy protection — the correspondence between the nodes of different graphs is unknown and accordingly, the graphs are unaligned. Additionally, the graph is generated from the most recent historical observations, *i.e.*, the last K observed graphs, which is reasonable in practice. For example, in social networks like Twitter and Facebook, the registration and the cancellation of user accounts and the interactions between users are often predictable based on historical observations.

Learning this model requires us to jointly *align* arbitrary-sized observed graphs and *predict* future graphs based on the observations. To deal with arbitrary-sized and unaligned graphs, we design our model as a composition of two autoregressions orienting to the number of nodes and the graph structure, respectively, as shown in Figure 1. Decoupling the target graph autoregressive model in (1) into two autoregressions simplifies the model complexity greatly. The size-oriented autoregression estimates the number of nodes for the target graph, and the structure-oriented autoregression aligns historical graphs jointly and aggregates them to predict node and edge probabilities. In the following content, we will show that the structure-oriented autoregression corresponds to a FGW barycenter of graphs (Vayer et al. 2020), whose learning and approximation can be achieved efficiently based on the proposed IFGW distance.

Two Autoregressions for Graph Generation

FGW distance for graph alignment Denote each graph G as a tuple $(N, \boldsymbol{\mu}, \boldsymbol{A}, \boldsymbol{V})$, where N is the number of nodes, the vector $\boldsymbol{\mu}$ in the $(N-1)$-Simplex Δ^{N-1} represents the empirical node distribution,[1] $\boldsymbol{A} = [a_{ij}] \in \mathbb{R}^{N\times N}$ is the adjacency matrix, and $\boldsymbol{V} \in \mathbb{R}^{N\times D}$ represents D-dimensional node features.[2] Given two graphs $G_1 = (N_1, \boldsymbol{\mu}_1, \boldsymbol{A}_1, \boldsymbol{V}_1)$ and $G_2 = (N_2, \boldsymbol{\mu}_2, \boldsymbol{A}_2, \boldsymbol{V}_2)$, the FGW distance (Vayer et al. 2020) between them is

$$\begin{aligned} d_{\text{fgw}}(G_1, G_2) = {} & \min_{\boldsymbol{T}\in\Pi_{\boldsymbol{\mu}_1,\boldsymbol{\mu}_2}} \mathbb{E}_{i,j\sim\boldsymbol{T}}[d(\boldsymbol{v}_i^1, \boldsymbol{v}_j^2)] + \\ & \mathbb{E}_{i,j,i',j'\sim\boldsymbol{T}\otimes\boldsymbol{T}}[L(a_{ii'}^1, a_{jj'}^2)] \\ = {} & \min_{\boldsymbol{T}\in\Pi_{\boldsymbol{\mu}_1,\boldsymbol{\mu}_2}} \sum_{i=1}^{N_1}\sum_{j=1}^{N_2} d(\boldsymbol{v}_i^1, \boldsymbol{v}_j^2) T_{ij} + \\ & \sum_{i,i'=1}^{N_1}\sum_{j,j'=1}^{N_2} L(a_{ii'}^1, a_{jj'}^2) T_{ij} T_{i'j'}, \end{aligned} \tag{2}$$

where $d(\boldsymbol{v}_i^1, \boldsymbol{v}_j^2)$ measures the distance between node features and $L(a_{ii'}^1, a_{jj'}^2)$ measures the distance between edges. In this work, we implement $d(\cdot,\cdot)$ as the cross-entropy loss for categorical node features (one-hot vectors) and mean-square-error (MSE) for continuous node features. For $L(\cdot,\cdot)$, we implement it as the cross-entropy loss for binary adjacency matrices and MSE for weighted adjacency matrices. $\boldsymbol{T} \in \Pi_{\boldsymbol{\mu}_1,\boldsymbol{\mu}_2}$, and $\Pi_{\boldsymbol{\mu}_1,\boldsymbol{\mu}_2} = \{\boldsymbol{T} \geq \boldsymbol{0} | \boldsymbol{T}\boldsymbol{1}_{N_1} = \boldsymbol{\mu}_2, \boldsymbol{T}^\top \boldsymbol{1}_{N_2} = \boldsymbol{\mu}_1\}$ is the set of node joint distributions with marginals $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$.

The FGW distance in (2) minimizes the expected loss of node pairs and that of edge pairs jointly, in which the first term corresponds to the Wasserstein distance (Cuturi 2013) between node features while the second term corresponds to the Gromov-Wasserstein (GW) distance between adjacency matrices (Mémoli 2011; Chowdhury and Mémoli 2019). The optimal $\boldsymbol{T}$ corresponding to the FGW distance is called optimal transport matrix, which corresponds to the optimal joint distribution of the graphs' nodes. In particular, the optimal transport achieves a probabilistic alignment of the graphs: the element T_{ij} can be interpreted as the probability that the node i in G_1 matches with the node j in G_2. Accordingly, $\boldsymbol{T} \otimes \boldsymbol{T}$ is a joint distribution of edges, where $\otimes$ is the Kronecker product. Its element $T_{ij}T_{i'j'}$ is the probability that the edge (i, i') is paired to the edge (j, j').

Given K graphs $\{G_k = (N_k, \boldsymbol{\mu}_k, \boldsymbol{A}_k, \boldsymbol{V}_k)\}_{k=1}^{K}$, we can define their N-node FGW barycenter (Peyré, Cuturi, and Solomon 2016; Vayer et al. 2020) as[3]

$$\bar{G}, \{\boldsymbol{T}_k\}_{k=1}^{K} = \arg\min_G \sum_{k=1}^{K} d_{\text{fgw}}(G, G_k), \tag{3}$$

where the barycenter $\bar{G} = (N, \boldsymbol{\mu}, \bar{\boldsymbol{A}}, \bar{\boldsymbol{V}})$ minimizes its FGW distances to the observed graphs and each $\boldsymbol{T}_k$ is the corresponding optimal transport between $\bar{G}$ and the G_k. Here, $\boldsymbol{\mu} = \frac{1}{N}\boldsymbol{1}_N$. The optimal transports and the barycenter are calculated by alternating optimization: the optimal transports are derived by computing the FGW distances based on current barycenter, and the new barycenter is obtained by

$$\bar{\boldsymbol{A}} = \frac{1}{\boldsymbol{\mu}\boldsymbol{\mu}^\top} \sum_{k=1}^{K} \boldsymbol{T}_k \boldsymbol{A}_k \boldsymbol{T}_k^\top, \ \bar{\boldsymbol{V}} = \frac{1}{\boldsymbol{\mu}\boldsymbol{1}_D^\top} \sum_{k=1}^{K} \boldsymbol{T}_k \boldsymbol{V}_k.$$

The FGW barycenter achieves a joint alignment of $\{G_k\}_{k=1}^{K}$ — for each G_k, its nodes are matched to the nodes of the barycenter (Xu, Luo, and Carin 2019). Compared with classic graph matching methods like quadratic assignment programming (QAP), the above FGW-based matching method has advantages on both robustness and efficiency.

Size-oriented autoregression Given $\{G_t\}_{t=0}^{T}$, we first focus on the sequence of graph sizes, *i.e.*, $\{N_t\}_{t=0}^{T}$, and model the dynamics of N_t via a classic K-order autoregressive model, denoted as $AR_N(K)$: $N_t = \sum_{k=1}^{K} \varphi_k N_{t-k} + \epsilon_t$, where $\boldsymbol{\varphi} = [\varphi_k]$ represents the coefficients of the model and ϵ_t represents the random noise imposed on the output. This model helps to predict the graph size of future graphs.[4] When assuming ϵ_t to be Gaussian, we can learn the coefficients by least squares estimation (LSE) (Hamilton 1994).

[1] In this work, we use a uniform distribution, *i.e.*, $\boldsymbol{\mu} = \frac{1}{N}\boldsymbol{1}_N$, but other node distributions are applicable as well.

[2] For a non-attributed graph, we treat $\boldsymbol{\mu}$ as its node features.

[3] In the following content, we ignore $\bar{G}$ in some equations for convenience because we leverage $\boldsymbol{T}$'s, rather than $\bar{G}$, in our model.

[4] When predicting graph sizes, we round the the output of the model to get integer numbers.

Structure-oriented autoregression Once the graph size N_t is determined, a structure-oriented K-order autoregression, denoted as $AR_S(K)$, predicts the isomorphism of G_t based on the last K historical graphs, *i.e.*, $\{G_{t-k} = (N_{t-k}, \boldsymbol{\mu}_{t-k}, \boldsymbol{A}_{t-k}, \boldsymbol{V}_{t-k})\}_{k=1}^K$. For the graphs with binary adjacency matrices and categorical node features, we formulate their autoregressive process by:

$$
\begin{aligned}
&G_t = (N_t, \boldsymbol{\mu}_t, \boldsymbol{A}_t, \boldsymbol{V}_t) \cong \widehat{G}_{t;g_1,g_2} = (N_t, \boldsymbol{\mu}_t, \widehat{\boldsymbol{A}}_t, \widehat{\boldsymbol{V}}_t),\\
&N_t \sim AR_N(K),\ \boldsymbol{\mu}_t = \frac{1}{N_t}\mathbf{1}_{N_t}\\
&\boldsymbol{A}_t, \boldsymbol{V}_t \cong \widehat{\boldsymbol{A}}_t \sim \text{Bernoulli}(\boldsymbol{P}_t^1),\ \widehat{\boldsymbol{V}}_t \sim \text{Categorical}(\boldsymbol{P}_t^2)\\
&\boldsymbol{P}_t^1 = g_1(\{\boldsymbol{A}_{t-k}\}_{k=1}^K; N_t, \{\boldsymbol{T}_{t,k}\}_{k=1}^K)\\
&\quad = \text{sigmoid}(\text{MLP}(\text{CAT}(\{\boldsymbol{T}_{t,k}\boldsymbol{A}_{t-k}\boldsymbol{T}_{t,k}^\top\}_{k=1}^K)))\\
&\boldsymbol{P}_t^2 = g_2(\{\boldsymbol{V}_{t-k}\}_{k=1}^K; N_t, \{\boldsymbol{T}_{t,k}\}_{k=1}^K)\\
&\quad = \text{softmax}(\text{MLP}(\text{CAT}(\{\boldsymbol{T}_{t,k}\boldsymbol{V}_{t-k}\}_{k=1}^K))),
\end{aligned}
\tag{4}
$$

where $\widehat{G}_{t;g_1,g_2}$ is an isomorphism of G_t generated by $AR_S(K)$, which is parameterized by two neural networks g_1 and g_2. $\boldsymbol{A}_t$ and $\boldsymbol{V}_t$ are the binary adjacency matrix and the one-hot node feature matrix of G_t, while $\widehat{\boldsymbol{A}}_t$ and $\widehat{\boldsymbol{V}}_t$ are their isomorphic estimations, which are sampled from a Bernoulli distribution and a categorical distribution, respectively. The parameters of these two distributions, *i.e.*, $\boldsymbol{P}_t^1 \in [0,1]^{N_t \times N_t}$ and $\boldsymbol{P}_t^2 \in \Delta^{N_t\times(D-1)}$,[5] are parametrized by two neural networks g_1 and g_2. We implement them by stacking concatenation (CAT) layer, multi-layer perceptron (MLP), and nonlinear activations. They take the historical graphs as their inputs, whose hyperparameters are the graph size N_t and the node joint distributions $\{\boldsymbol{T}_{t,k}\}_{k=1}^K$.

After aligning $\{\boldsymbol{A}_{t-k}, \boldsymbol{V}_{t-k}\}_{k=1}^K$ as $\{\boldsymbol{T}_{t,k}\boldsymbol{A}_{t-k}\boldsymbol{T}_{t,k}^\top\}_{k=1}^K$ and $\{\boldsymbol{T}_{t,k}\boldsymbol{V}_{t-k}\}_{k=1}^K$, g_1 concatenates $\{\boldsymbol{T}_{t,k}\boldsymbol{A}_{t-k}\boldsymbol{T}_{t,k}^\top\}_{k=1}^K$ to a tensor in $\mathbb{R}^{N_t\times N_t\times K}$, then maps the tensor to a matrix in $\mathbb{R}^{N_t\times N_t}$ by MLP, and uses the sigmoid function to ensure the output in $[0,1]^{N_t\times N_t}$. g_2 applies a similar pipeline to $\{\boldsymbol{T}_{t,k}\boldsymbol{V}_{t-k}\}_{k=1}^K$. Besides the graphs with binary adjacency matrices and categorical node features, our model is applicable to other kinds of graphs — just replacing the distributions and the activations in (4) with other configurations.

Naive Learning Paradigm: Bi-level Optimization

The combination of above two autoregressions leads to our GWAR model with parameters $\boldsymbol{\varphi}$, g_1, and g_2. Given a sequence of observed graphs $\{G_t\}_{t=0}^T$, we learn $\boldsymbol{\varphi}$ by the classic least-square method in (Hamilton 1994) and learn g_1 and g_2 by solving the following bi-level optimization problem.

$$
\begin{aligned}
&\min_{g_1,g_2} \sum\nolimits_{t=K+1}^{T} d_{\text{fgw}}(\widehat{G}_{t;g_1,g_2}, G_t),\\
&s.t.\ \forall t, \{\boldsymbol{T}_{t,k}\}_{k=1}^K \in \arg\min \sum\nolimits_{k=1}^{K} d_{\text{fgw}}(G, G_{t-k}),
\end{aligned}
\tag{5}
$$

where $\widehat{G}_{t;g_1,g_2} = (N_t, \boldsymbol{\mu}_t, \boldsymbol{P}_t^1, \boldsymbol{P}_t^2)$ is the probabilistic estimation of G_t's isomorphism, and the upper-level problem aims to minimize the FGW distance between G_t and $\widehat{G}_{t;g_1,g_2}$. Essentially, this objective function pursues the structural similarity between each observation and its prediction — the FGW distance (Vayer et al. 2020; Chowdhury and Mémoli 2019) can be valid metrics of graph, which are equal to zero for isomorphic graphs. The lower-level problem computes the FGW barycenter. As shown in (3), solving this problem leads to the joint alignment of historical graphs.

Note that the optimization problem in (5) is statistically-interpretable. In particular, when we implement the node pair loss d and the edge pair loss L in (2) as cross-entropy losses, the objective function can be rewritten as

$$
\begin{aligned}
\min_{\boldsymbol{T}\in\Pi_{\boldsymbol{\mu}_t,\boldsymbol{\mu}_t}} &\mathbb{E}_{i,j\sim\boldsymbol{T}}[\text{NLL}(\boldsymbol{p}_{jt}^2; \boldsymbol{v}_{it})] +\\
&\mathbb{E}_{i,j,i',j'\sim\boldsymbol{T}\otimes\boldsymbol{T}}[\text{NLL}(p_{jj't}^1; a_{ii't})],
\end{aligned}
\tag{6}
$$

where $p_{jj't}^1$ is the element of $\boldsymbol{P}_t^1$, $\boldsymbol{p}_{jt}^2$ is the j-th row of $\boldsymbol{P}_t^2$. $\text{NLL}(p_{jj'}^1; a_{ii'}) = -a_{ii't}\log p_{jj't}^1 - (1-a_{ii't})\log(1-p_{jj't}^1)$ is the negative log-likelihood of the edge probability $p_{jj't}^1$ under the condition that the edge (i,i') in G_t matches with the edge (j,j') in $\widehat{G}_{t;g_1,g_2}$. $\text{NLL}(\boldsymbol{p}_{jt}^2; \boldsymbol{v}_{it}) = -\boldsymbol{v}_{it}^\top \log \boldsymbol{p}_{jt}^2$ is negative log-likelihood of the node feature probability vector $\boldsymbol{p}_{jt}^2$ under the condition that the i-th node in G_t matches with the j-th node in $\widehat{G}_{t;g_1,g_2}$. Therefore, the optimal $\boldsymbol{T}$ leads to the minimum expectation of the negative log-likelihoods.

Such a bi-level optimization problem can be solved approximately by an alternating optimization strategy (Xu 2020): given historical graphs, we can first estimate the $\boldsymbol{T}_{t,k}$'s by solving the lower-level problem and then update g_1 and g_2 by solving the upper-level problem. The upper-level problem also involves a nested alternating optimization — we need to calculate the $\boldsymbol{T}$ associated with $d_{\text{fgw}}(\widehat{G}_{t;g_1,g_2}, G_t)$ and then update g_1 and g_2 by gradient decent under fixed $\boldsymbol{T}$. Repeating the two steps till the upper-level objective function converges, we obtain the proposed GWAR model. Note that, when the g_1 and g_2 in (4) are linear models and their parameters are shared and in the Simplex, we can simplify the problem as follows:

Proposition 1. *In the case that*

$$
\begin{aligned}
&g_1(\{\boldsymbol{A}_{t-k}\}_{k=1}^K; N_t, \{\boldsymbol{T}_{t,k}\}_{k=1}^K) = N_t^2 \sum\nolimits_k \beta_k \boldsymbol{T}_{t,k}\boldsymbol{A}_{t-k}\boldsymbol{T}_{t,k}^\top\\
&g_2(\{\boldsymbol{V}_{t-k}\}_{k=1}^K; N_t, \{\boldsymbol{T}_{t,k}\}_{k=1}^K) = N_t \sum\nolimits_k \beta_k \boldsymbol{T}_{t,k}\boldsymbol{V}_{t-k},
\end{aligned}
$$

where $\boldsymbol{\beta} = [\beta_k] \in \Delta^{K-1}$ is the model parameter. We can simplify the problem (5) *and solve it approximately as*

$$
\begin{aligned}
&\min_{\boldsymbol{\beta}} \sum\nolimits_{t=K+1}^{T} NLL(\boldsymbol{P}_t^1; \boldsymbol{A}_t) + NLL(\boldsymbol{P}_t^2; \boldsymbol{V}_t),\\
&s.t.\ \forall t, \{\boldsymbol{T}_{t,k}\}_{k=1}^K \in \arg\min \sum\nolimits_{k=1}^{K} \beta_k d_{fgw}(G, G_{t-k}),
\end{aligned}
\tag{7}
$$

Proof. In this case, $\boldsymbol{P}_t^1 = g_1(\{\boldsymbol{A}_{t-k}\}_{k=1}^K; N_t, \{\boldsymbol{T}_{t,k}\}_{k=1}^K)$ and $\boldsymbol{P}_t^2 = g_2(\{\boldsymbol{V}_{t-k}\}_{k=1}^K; N_t, \{\boldsymbol{T}_{t,k}\}_{k=1}^K)$, which coincident with a weighted FGW barycenter (Xu 2020):

$$
\bar{G}_t, \{\boldsymbol{T}_{t,k}\}_{k=1}^K \in \arg\min \sum\nolimits_{k=1}^{K} \beta_k d_{\text{fgw}}(G, G_{t-k}),
\tag{8}
$$

where the $\bar{\boldsymbol{A}}_t$ and $\bar{\boldsymbol{V}}_t$ of $\bar{G}_t$ are $\boldsymbol{P}_t^1$ and $\boldsymbol{P}_t^2$, respectively. When $\boldsymbol{P}_t^1, \boldsymbol{P}_t^2 \in \arg\min \sum_{k=1}^{K} \beta_k d_{\text{fgw}}(G, G_{t-k})$, their

[5] $\Delta^{N_t\times(D-1)}$ means the set of the matrices of size $N_t \times D$, and each row of such matrices is in the $(D-1)$-Simplex.

isomorphisms, $i.e.$, $R(\boldsymbol{P}_t^1), R(\boldsymbol{P}_t^2)$ with an arbitrary permutation R imposed on the node indices, are also the optimal solution of (8). In this case, the optimal transport $\boldsymbol{T}$ in the upper-level objective (6) becomes redundant (Xu 2020) because the $\boldsymbol{T}$ corresponds to a permutation of node indices as well. Therefore, removing the optimal transport problem in the upper-level objective (6) and applying (8) as the lower-level problem, we simplify (5) as (7). □

This proposition indicates that in this special case, our GWAR model becomes a mixture model of historical graphs under the FGW distance. Accordingly, the alignment of historical graphs and the estimation of the new graph can be achieved jointly by computing a FGW barycenter.

Unfortunately, both (5) and its simplified version (7) require to compute the FGW barycenter when solving the lower-level problem. Moreover, because the alignment of historical graphs is necessary for both learning and inference, the computation of the barycenter is required in the testing phase. In particular, this step involves computing K FGW distances iteratively, which limits the application of our GWAR model in practice because the complexity of FGW distance is $\mathcal{O}(N^3)$ in general for the graphs with N nodes. What is worse, it not only owns time-consuming feed-forward computations but also leads to complicated backpropation — computing the gradients of $\boldsymbol{T}$'s by AutoDiff is slow and costs a lot of memory because of the iterative computation of the $\boldsymbol{T}$'s. Although some attempts have been made to reduce the complexity of the FGW distance, $e.g.$, the scalable GWL in (Xu, Luo, and Carin 2019), the low-rank GW distance in (Scetbon, Peyré, and Cuturi 2021), and so on, they still need to compute the optimal transport matrix in a nonparametric and transductive way. To make our GWAR model applicable in practice, we parametrize the optimal transport and propose a learning-based inductive FGW distance, which reformulates the learning task and reduces its computational cost greatly.

Efficient Learning and Inference

Inductive FGW Distance

According to the above analysis, the main bottleneck of our model is the computation of the FGW distance and barycenters. To reduce its computational complexity, we design an inductive FGW (IFGW) distance. Our design is inspired by the amortization strategy of variational inference (Kim et al. 2018), which parametrizes the variational probability of each sample by a shared encoder and takes the samples as the inputs. Given arbitrary two graphs G_1 and G_2, we parameterize the optimal transport associated with their FGW distance by a graph neural network (GNN) with coupled attention layers. As shown in Figure 2(a), GNN converts the two graphs to two sets of node embeddings, $i.e.$, $\boldsymbol{X}_1 = h(G_1) \in \mathbb{R}^{N_1\times M}$ and $\boldsymbol{X}_2 = h(G_2) \in \mathbb{R}^{N_2\times M}$, where we set h to be a graph isomorphism network (GIN) (Xu et al. 2018) in this work. Assuming the node embeddings to yield a normal prior distribution, we derive two attention maps:

$$\begin{aligned}\boldsymbol{S} &= \text{diag}(\boldsymbol{\mu}_1)\text{Gumbel-sm}_{\text{row}}(\boldsymbol{X}_1\boldsymbol{W}_1\boldsymbol{W}_2^\top\boldsymbol{X}_2^\top),\\ \boldsymbol{Z} &= \text{Gumbel-sm}_{\text{col}}(\boldsymbol{X}_1\boldsymbol{U}_1\boldsymbol{U}_2^\top\boldsymbol{X}_2^\top)\text{diag}(\boldsymbol{\mu}_2),\end{aligned} \tag{9}$$

where $\boldsymbol{\gamma} = \{\boldsymbol{W}_1, \boldsymbol{W}_2, \boldsymbol{U}_1, \boldsymbol{U}_2 \in \mathbb{R}^{M\times H}\}$ are linear mappings, and Gumbel-sm$_{\text{row}}$ and Gumbel-sm$_{\text{col}}$ apply row-wise and column-wise Gumbel-softmax (Jang, Gu, and Poole 2016) to a matrix, respectively. Using Gumbel-softmax can suppress the over-smoothness problem when the number of nodes is large. Obviously, $\boldsymbol{S} \in \Pi_{\boldsymbol{\mu}_1,\cdot}$ and $\boldsymbol{Z} \in \Pi_{\cdot,\boldsymbol{\mu}_2}$, so that when $\boldsymbol{S} = \boldsymbol{Z}$, we obtain a matrix in $\Pi_{\boldsymbol{\mu}_1,\boldsymbol{\mu}_2}$.

In summary, our IFGW distance decouples the two-side marginal constraint on the optimal transport matrix $\boldsymbol{T}$ to two one-side constraints on $\boldsymbol{S}$ and $\boldsymbol{Z}$. We learn the parameters of neural networks by solving the optimization problem:

$$\begin{aligned}&\min_{h,\boldsymbol{\gamma}} \text{loss}(G_1, G_2; h, \boldsymbol{\gamma}) =\\ &\min_{h,\boldsymbol{\gamma}} \underbrace{\mathbb{E}_{i,j\sim\boldsymbol{S}}[d(\boldsymbol{v}_i^1, \boldsymbol{v}_j^2)] + \mathbb{E}_{i,i',j,j'\sim\boldsymbol{S}\otimes\boldsymbol{Z}}[L(a_{ii'}^1, a_{jj'}^2)]}_{d_{\text{ifgw}}(G_1,G_2;h,\boldsymbol{\gamma})}\\ &\quad + \tau_1\text{KL}(\boldsymbol{S}\|\boldsymbol{Z}) + \tau_2(\text{KL}(\boldsymbol{X}_1\|\mathcal{N}) + \text{KL}(\boldsymbol{X}_2\|\mathcal{N})),\end{aligned} \tag{10}$$

where $\boldsymbol{S} = \boldsymbol{S}(G_1, G_2; h, \boldsymbol{\gamma})$ and $\boldsymbol{Z} = \boldsymbol{Z}(G_1, G_2; h, \boldsymbol{\gamma})$ are neural networks taking G_1 and G_2 as their inputs and with parameters $h, \boldsymbol{\gamma}$. In (10), the first line of objective function corresponds to the definition of our IFGW distance, while the second line contains the regularizers ensuring i) $\boldsymbol{S}$ and $\boldsymbol{Z}$ are similar to each other; and ii) the distributions of the node embeddings are close to a normal distribution $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_M)$.

After solving (10), given arbitrary two graphs, we can estimate their discrepancy by the learned IFGW distance, in which the optimal transport is derived directly by (9) based on the learned neural network. Because of avoiding iterative optimization, the IFGW distance is extremely fast. Based on the IFGW distance, we can further define the IFGW barycenter and approximate multiple optimal transports in an inductive way. Recall that the output of GIN yields a normal prior distribution. Therefore, instead of computing the barycenter graph explicitly, we sample from the normal distribution and approximate the embeddings of the barycenter's nodes directly. As a result, given $\{G_k\}_{k=1}^K$, we obtain a N-size IFGW barycenter and the optimal transports as

$$\begin{aligned}&\bar{G}, \{\boldsymbol{T}_k\}_{k=1}^K = \arg\min \textstyle\sum_{k=1}^K d_{\text{ifgw}}(G, G_k; h, \boldsymbol{\gamma})\\ \Leftrightarrow\ &\boldsymbol{\mu} = \frac{1}{N}\boldsymbol{1}_N,\ \bar{\boldsymbol{X}} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_M),\ \boldsymbol{X}_k = h(G_k)\\ &\boldsymbol{S}_k = \text{diag}(\boldsymbol{\mu})\text{Gumbel-sm}_{\text{row}}(\boldsymbol{X}\boldsymbol{W}_1\boldsymbol{W}_2^\top\boldsymbol{X}_k^\top)\\ &\bar{\boldsymbol{A}} = \frac{1}{\boldsymbol{\mu}\boldsymbol{\mu}^\top}\sum_k \boldsymbol{S}_k\boldsymbol{A}_k\boldsymbol{S}_k^\top,\ \bar{\boldsymbol{V}} = \frac{1}{\boldsymbol{\mu}\boldsymbol{1}_D^\top}\sum_k \boldsymbol{S}_k\boldsymbol{V}_k.\end{aligned} \tag{11}$$

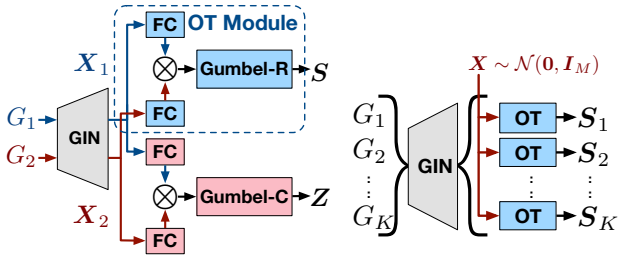
Figure 2(b) shows the computation of $\{\boldsymbol{S}_k\}_{k=1}^K$.

Two-Step Optimization of Our GWAR Model

Replacing the FGW distance with the IFGW distance, our learning task is reformulated as two optimization problems:

Learning the IFGW distance Sample arbitrary two graphs from the graph sequence and solve

$$\min_{h,\boldsymbol{\gamma}} \textstyle\sum_{G,G'\in\{G_t\}_{t=0}^T} \text{loss}(G, G'; h, \boldsymbol{\gamma}). \tag{12}$$



(a) IFGW distance (b) IFGW barycenter

Figure 2: (a) An illustration of IFGW distance, where "$\otimes$" means inner product and FC means linear layer. The OT module in the dotted frame infers optimal transport. (b) An illustration of IFGW barycenter, which leverages a GIN and the OT module to infer multiple optimal transports.

Learning the GWAR model Fix the learned parameters $\{h^*, \boldsymbol{\gamma}^*\}$ and optimize the following problem:

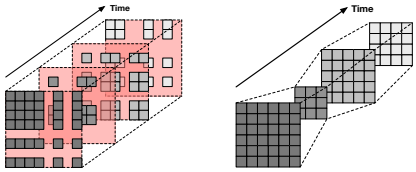
$$\begin{aligned} &\min_{g_1,g_2} \sum_{t=K+1}^{T} d_{\text{ifgw}}(\widehat{G}_{t;g_1,g_2}, G_t; h^*, \boldsymbol{\gamma}^*), \\ &s.t.\ \forall t, \\ &\{\boldsymbol{T}_{t,k}\}_{k=1}^{K} = \arg\min \sum_k d_{\text{ifgw}}(G, G_{t-k}; h^*, \boldsymbol{\gamma}^*). \end{aligned} \tag{13}$$

(13) is not a bi-level optimization because $\boldsymbol{T}_{t,k}$'s can be derived by (11) directly. We solve (12) and (13) by stochastic gradient descent and obtain the GWAR model accordingly. The learned IFGW distance is applicable in the inference phase so it also accelerates our model when predicting.

Experiments

We apply our **GWAR** to both synthetic and real-world graph modeling tasks, and compare it with the state-of-the-art models, including **VGRNN** (Hajiramezanali et al. 2019), **DynAE**, **DynRNN** and **DynAERNN** (Goyal, Chhetri, and Canedo 2020). Specifically, DynAE is a graph autoencoder. DynRNN and DynAERNN are based on recurrent neural networks. VGRNN models the hidden states as a graph recurrent neural network. When implementing our GWAR, we set the number of GIN layers to be 5, the dimension $M = 64$, the dimension $H = 32$, and the weights of regularizers in (10) are $\tau_1 = \tau_2 = 0.1$. For the g_1 and g_2 of GWAR, their MLPs contains two hidden layers, whose hidden dimension is 32. We learn both (12) and (13) by SGD and set the batch size to be 5, the learning rate to be 0.05, and the number of epochs to be 100. We run the experiments below on a server with two Nvidia GTX3090 GPUs.

For each method, we consider one synthetic dataset and three real-world datasets. **Synthetic graph sequence:** We generate a sequence of synthetic graphs with length $T = 100$ by a 5-order structure-oriented autoregression, where the size of each graph is fixed as 50, the T's in (4) are identity matrices, and the remaining model parameters and the initial five graphs are set randomly. Each graph contains a binary adjacency matrix, uniform node distribution, and one-hot node features. **Email-EU:** The Email-EU temporal network in (Paranjape, Benson, and Leskovec 2017)



(a) Baselines (b) GWAR

Figure 3: The comparison for various methods on their learning strategies. The baselines require the information of node indices while our GWAR does not require it.

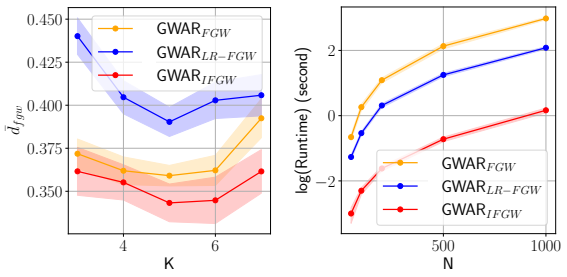
is a collection of emails between members of a European research institution. We extract a sequence of graphs, in which each graph represents the email interactions per day. **UCI:** The UCI dataset (Panzarasa, Opsahl, and Carley 2009) comprises of private messages sent on an online social network at the University of California, Irvine. The graph sequence contains social interactions per day. **Math-Small:** The Math-Overflow network in (Paranjape, Benson, and Leskovec 2017) is a collection of interactions on the stack exchange website. Three types of interactions are included: answers to questions, comments to questions and comments to answers. We sample a subset called Math-Small, which contains one graph per month and ensures each graph has at least 30 edges and at most 3,000 nodes.

For each dataset, we imitate unaligned scenarios by merely preserving $\Delta N \times 100\%$ nodes randomly from each graph, where $\Delta N \sim \text{Uniform}(\alpha - 0.1, \alpha + 0.1)$ and $\alpha \in \{0.9, 0.7, 0.5\}$ controls the seriousness of unalignment. Given such unaligned evolutionary graphs, the baselines have to use the indices of the preserved nodes and treat the learning task as predicting links with missing data. Note that without the side information of node indices, all the baselines are inapplicable, while GWAR does not require such side information, which can be trained directly on the sequences of arbitrary-sized and unaligned graphs. Figure 3 shows the difference between our models and the baselines.

For each dataset, we convert the long unaligned graph sequence to N short sequences of length $K + 1$, where the order $K = 5$ for the synthetic data and 7 (the number of days per week) for the real-world data. For each model, we train it on 80% sequences and test it on the remaining 20% sequences by predicting the last graph based on the K observations. Our GWAR only generates the isomorphisms of target graphs because it does not leverage any side information of node index. Therefore, for fairness, we evaluate the performance of each model by measuring the average of the FGW distance between each real graph and the corresponding generated graph, *i.e.*, $\bar{d}_{\text{fgw}} = \frac{1}{N}\sum_{n=1}^{N} d_{\text{fgw}}(G_n, \widehat{G}_n)$, which reflects the accuracy of the structural information captured by the models. For the graphs without node attributes, we just compare the estimated adjacency matrices with the real ones and the measurement becomes the averaged GW distance (Chowdhury and Mémoli 2019), *i.e.*, $\bar{d}_{\text{gw}}$. Besides $\bar{d}_{\text{gw}}$, we match the predicted graph achieved by each method with the ground truth and calculate the F1-score of the link

Sampling rate α	0.9	0.7	0.5
VGRNN	$0.447_{\pm 0.012}$	$0.452_{\pm 0.017}$	$0.466_{\pm 0.023}$
DynAE	$0.480_{\pm 0.018}$	$0.507_{\pm 0.024}$	$0.520_{\pm 0.035}$
DynRNN	$0.496_{\pm 0.019}$	$0.509_{\pm 0.027}$	$0.525_{\pm 0.032}$
DynAERNN	$0.470_{\pm 0.011}$	$0.479_{\pm 0.015}$	$0.485_{\pm 0.020}$
GWAR_{FGW}	$0.384_{\pm 0.004}$	$0.438_{\pm 0.008}$	$0.460_{\pm 0.010}$
$\text{GWAR}_{\text{LR-FGW}}$	$0.356_{\pm 0.006}$	$0.441_{\pm 0.009}$	$0.473_{\pm 0.011}$
$\text{GWAR}_{\text{IFGW}}$	$\mathbf{0.348}_{\pm 0.012}$	$\mathbf{0.385}_{\pm 0.019}$	$\mathbf{0.406}_{\pm 0.021}$

Table 1: Comparisons on synthetic data ($\bar{d}_{\text{fgw}} \pm$Std.)



(a) $\bar{d}_{\text{fgw}}$ w.r.t. K (b) $\log_{10}$(Time) w.r.t. N

Figure 4: (a) The performance under various K's. (b) The runtime per batch with respect to different graph sizes.

Dataset	Email-EU	UCI	Math-Small
Time Span	802 days	193 days	78 months
#Nodes	89	1899	3000
#Total Edges	12216	17950	50652
VGRNN	$0.481_{\pm 0.013}$	$0.755_{\pm 0.006}$	Out-of-memory
DynAE	$0.475_{\pm 0.028}$	$0.700_{\pm 0.001}$	$0.787_{\pm 0.003}$
DynRNN	$0.507_{\pm 0.031}$	$0.760_{\pm 0.017}$	Out-of-memory
DynAERNN	$0.493_{\pm 0.019}$	$0.735_{\pm 0.002}$	$0.801_{\pm 0.005}$
GWAR_{FGW}	$0.424_{\pm 0.009}$	$0.669_{\pm 0.014}$	Out-of-memory
$\text{GWAR}_{\text{LR-FGW}}$	$0.460_{\pm 0.013}$	$0.678_{\pm 0.017}$	Out-of-memory
$\text{GWAR}_{\text{IFGW}}$	$\mathbf{0.399}_{\pm 0.016}$	$\mathbf{0.652}_{\pm 0.020}$	$\mathbf{0.763}_{\pm 0.022}$

Table 2: Comparisons on real-world data ($\bar{d}_{\text{gw}} \pm$ Std.)

	Synthetic	Email-EU	UCI
VGRNN	0.829	0.693	0.658
DynAE	0.832	0.702	0.680
DynRNN	0.831	0.713	**0.684**
DynAERNN	0.829	0.690	0.662
GWAR_{FGW}	0.838	0.706	0.677
$\text{GWAR}_{\text{LR-FGW}}$	0.825	0.698	0.678
$\text{GWAR}_{\text{IFGW}}$	**0.840**	**0.715**	**0.684**

Table 3: Comparisons for various methods on F1-score

prediction results. For our GWAR, the F1-score is calculated based on the optimal transport matrix associated with $\bar{d}_{\text{gw}}$, while for the baselines, the F1-score is derived directly based on the link prediction results. To demonstrate the efficiency of our IFGW distance, we compare the runtime and performance of our learning method with the IFGW distance, the original FGW distance (Vayer et al. 2020), and the low-rank FGW (LR-FGW) distance implemented based on (Scetbon, Peyré, and Cuturi 2021). We train and test each model in 5 trials and record the mean and the standard deviation.

Table 1 lists the learning results derived by different models. We can find that our GWAR outperforms its competitors consistently, which verifies its superiority. When increasing the noise (desceasing α), although the performance of our GWAR degrades slightly, its results are still better than those of the baselines. In particular, the learning of the baselines requires well-aligned graphs. Accordingly, the noise imposed on the data will break the evolutionary patterns of edges directly (as shown in Figure 3(a)). On the contrary, our GWAR fully leverages all edges of historical graphs to predict each future edge because the soft alignment achieved by the $\boldsymbol{T}$'s. In other words, our model considers more information and thus mitigates the negative influence of noise.

To demonstrate the robustness of our method to the model misspecification problem, we design several GWAR models with different K's and train the model on the synthetic data. Figure 4(a) shows that the performance of our GWAR is relatively stable with respect to the change of K's when it is learned based on the IFGW distance, which means that our methods are robust to the model misspecification problem. Figure 4(b) visualizes the GWAR on the runtime per graph. The result shows that applying our IFGW distance indeed improves the efficiency of our model greatly, which achieves about 100 times acceleration compared with the models using FGW or LR-FGW distances in the case of large graphs.

Table 2 shows that our GWAR achieves better results than the baselines on the three real-world datasets, which further shows the effectiveness of our model. Moreover, when applying our IFGW distance, we improve the scalability of GWAR greatly. In particular, for the Math-Small dataset, GWAR with the original FGW distance and its low-rank version suffer from the out-of-memory issue caused by the iterative optimization of optimal transports. Some baselines also have the same problem in our machine. Applying the IFGW distance helps our model to avoid the issue successfully and achieve the best performance. Besides calculating the FGW distance between each real graph and the corresponding generated graph, we can match the generated graph to the real one and calculate the F1-score of the link prediction result. Table 3 shows the comparisons of various methods on both synthetic and read-world datasets.

Conclusion

We have proposed a novel GWAR model for privacy-preserved evolutionary graph modeling. To learn and apply the model efficiently, we develop an acceleration strategy based on a novel IFGW distance and make the model applicable for large-scale graphs. Our GWAR makes the first attempt to build an autoregressive model for unaligned evolutionary graphs, which protects private information of nodes hidden behind their temporal behaviors. The IFGW distance is an efficient method that approximates optimal transports in a parametric and inductive way. In the future, we plan to improve our method for link prediction and community tracking in private-preserving scenarios.

Acknowledgments

Dr. Dixin Luo was supported in part by the National Natural Science Foundation of China (No. 62102031); the Young Scholar Program (No. XSQD-202107001) and the Technology Innovation Program (No. 3070012212116) from Beijing Institute of Technology, and the project No. 2020YFF0305200. Dr. Hongteng Xu was supported by the National Natural Science Foundation of China (No. 62106271), CAAI-Huawei MindSpore Open Fund, the Fundamental Research Funds for the Central Universities, and the Research Funds of Renmin University of China. He also would like to thank the supports from the Beijing Key Laboratory of Big Data Management and Analysis Methods.

References

Alvarez-Melis, D.; and Jaakkola, T. 2018. Gromov-Wasserstein Alignment of Word Embedding Spaces. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 1881–1890.

Bojchevski, A.; Shchur, O.; Zügner, D.; and Günnemann, S. 2018. Netgan: Generating graphs via random walks. In *International Conference on Machine Learning*, 610–619. PMLR.

Cheng, Z.; Caverlee, J.; and Lee, K. 2010. You are where you tweet: a content-based approach to geo-locating twitter users. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, 759–768.

Chowdhury, S.; and Mémoli, F. 2019. The gromov–wasserstein distance between networks and stable network invariants. *Information and Inference: A Journal of the IMA*, 8(4): 757–787.

Cuturi, M. 2013. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in neural information processing systems*, 2292–2300.

Durante, D.; and Dunson, D. B. 2014. Bayesian dynamic financial networks with time-varying predictors. *Statistics & Probability Letters*, 93: 19–26.

Dwork, C.; McSherry, F.; Nissim, K.; and Smith, A. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, 265–284. Springer.

Erdos, P.; Rényi, A.; et al. 1960. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1): 17–60.

Goyal, P.; Chhetri, S. R.; and Canedo, A. 2020. dyngraph2vec: Capturing network dynamics using dynamic graph representation learning. *Knowledge-Based Systems*, 187: 104816.

Hajiramezanali, E.; Hasanzadeh, A.; Narayanan, K.; Duffield, N.; Zhou, M.; and Qian, X. 2019. Variational Graph Recurrent Neural Networks. *Advances in Neural Information Processing Systems*, 32: 10701–10711.

Hamilton. 1994. *Time Series Analysis*. Time Series Analysis.

Holland, P. W.; Laskey, K. B.; and Leinhardt, S. 1983. Stochastic blockmodels: First steps. *Social networks*, 5(2): 109–137.

Igamberdiev, T.; and Habernal, I. 2021. Privacy-preserving graph convolutional networks for text classification. *arXiv preprint arXiv:2102.09604*.

Jang, E.; Gu, S.; and Poole, B. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Kim, Y.; Wiseman, S.; Miller, A.; Sontag, D.; and Rush, A. 2018. Semi-amortized variational autoencoders. In *International Conference on Machine Learning*, 2678–2687. PMLR.

Kumar, S.; Zhang, X.; and Leskovec, J. 2019. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1269–1278.

Li, X.; Zhou, Y.; Dvornek, N.; Zhang, M.; Gao, S.; Zhuang, J.; Scheinost, D.; Staib, L. H.; Ventola, P.; and Duncan, J. S. 2021. Braingnn: Interpretable brain graph neural network for fmri analysis. *Medical Image Analysis*, 74: 102233.

Li, Y.; Qian, B.; Zhang, X.; and Liu, H. 2020. Graph neural network-based diagnosis prediction. *Big Data*, 8(5): 379–390.

Liao, R.; Li, Y.; Song, Y.; Wang, S.; Hamilton, W.; Duvenaud, D. K.; Urtasun, R.; and Zemel, R. 2019. Efficient Graph Generation with Graph Recurrent Attention Networks. *Advances in Neural Information Processing Systems*, 32: 4255–4265.

Luo, D.; Xu, H.; Zha, H.; Du, J.; Xie, R.; Yang, X.; and Zhang, W. 2014. You are what you watch and when you watch: Inferring household structures from IPTV viewing data. *IEEE Transactions on Broadcasting*, 60(1): 61–72.

Mémoli, F. 2011. Gromov–Wasserstein distances and the metric approach to object matching. *Foundations of computational mathematics*, 11(4): 417–487.

Panzarasa, P.; Opsahl, T.; and Carley, K. M. 2009. Patterns and dynamics of users' behavior and interaction: Network analysis of an online community. *Journal of the American Society for Information Science and Technology*, 60(5): 911–932.

Paranjape, A.; Benson, A. R.; and Leskovec, J. 2017. Motifs in temporal networks. In *Proceedings of the tenth ACM international conference on web search and data mining*, 601–610.

Pareja, A.; Domeniconi, G.; Chen, J.; Ma, T.; Suzumura, T.; Kanezashi, H.; Kaler, T.; Schardl, T.; and Leiserson, C. 2020. Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 5363–5370.

Paul, M.; and Dredze, M. 2011. You are what you tweet: Analyzing twitter for public health. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 5, 265–272.

Peyré, G.; Cuturi, M.; and Solomon, J. 2016. Gromov-Wasserstein averaging of kernel and distance matrices. In *International Conference on Machine Learning*, 2664–2672. PMLR.

Qiu, Y.; Huang, C.; Wang, J.; Huang, Z.; and Xiao, J. 2022. A Privacy-Preserving Subgraph-Level Federated Graph Neural Network via Differential Privacy. *arXiv preprint arXiv:2206.03492*.

Scetbon, M.; Peyré, G.; and Cuturi, M. 2021. Linear-Time Gromov Wasserstein Distances using Low Rank Couplings and Costs. *arXiv preprint arXiv:2106.01128*.

Shen, Y.; Leus, G.; and Giannakis, G. B. 2019. Online graph-adaptive learning with scalability and privacy. *IEEE Transactions on Signal Processing*, 67(9): 2471–2483.

Simonovsky, M.; and Komodakis, N. 2018. Graphvae: Towards generation of small graphs using variational autoencoders. In *International conference on artificial neural networks*, 412–422. Springer.

Sinkhorn, R.; and Knopp, P. 1967. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21(2): 343–348.

Tantipathananandh, C.; Berger-Wolf, T.; and Kempe, D. 2007. A framework for community identification in dynamic social networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, 717–726.

Titouan, V.; Courty, N.; Tavenard, R.; and Flamary, R. 2019. Optimal transport for structured data with application on graphs. In *International Conference on Machine Learning*, 6275–6284. PMLR.

Vayer, T.; Chapel, L.; Flamary, R.; Tavenard, R.; and Courty, N. 2020. Fused Gromov-Wasserstein distance for structured objects. *Algorithms*, 13(9): 212.

Vayer, T.; Flamary, R.; Tavenard, R.; Chapel, L.; and Courty, N. 2019. Sliced gromov-wasserstein. *arXiv preprint arXiv:1905.10124*.

Villani, C. 2008. *Optimal transport: old and new*, volume 338. Springer Science & Business Media.

Wang, G.; Konolige, T.; Wilson, C.; Wang, X.; Zheng, H.; and Zhao, B. Y. 2013. You are how you click: Clickstream analysis for sybil detection. In *22nd USENIX Security Symposium (USENIX Security 13)*, 241–256.

Wang, J.; Zhang, S.; Xiao, Y.; and Song, R. 2021. A review on graph neural network methods in financial applications. *arXiv preprint arXiv:2111.15367*.

Wu, C.; Wu, F.; Cao, Y.; Huang, Y.; and Xie, X. 2021. Fedgnn: Federated graph neural network for privacy-preserving recommendation. *arXiv preprint arXiv:2102.04925*.

Xu, H. 2020. Gromov-Wasserstein factorization models for graph clustering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 6478–6485.

Xu, H.; Luo, D.; and Carin, L. 2019. Scalable gromov-Wasserstein learning for graph partitioning and matching. In *Advances in neural information processing systems*, 3046–3056.

Xu, H.; Luo, D.; Zha, H.; and Carin, L. 2019. Gromov-wasserstein learning for graph matching and node embedding. In *International conference on machine learning*, 6932–6941. PMLR.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*.

Zhang, L.; Zhao, L.; Qin, S.; Pfoser, D.; and Ling, C. 2021. TG-GAN: Continuous-time Temporal Graph Deep Generative Models with Time-Validity Constraints. In *Proceedings of the Web Conference 2021*, 2104–2116.

Zhou, D.; Zheng, L.; Han, J.; and He, J. 2020. A data-driven graph generative model for temporal interaction networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 401–411.