Learning Attribute-Structure Co-Evolutions in Dynamic Graphs

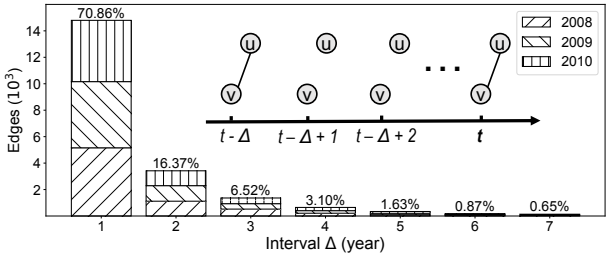
Daheng Wang, Zhihan Zhang, Yihong Ma, Tong Zhao,
Tianwen Jiang, Nitesh V. Chawla, Meng Jiang
Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556, USA
{dwang8,tzhao2,tjiang2,nchawla,mjiang2}@nd.edu
zhangzhihan@pku.edu.cn,yihongma97@gmail.com

ABSTRACT

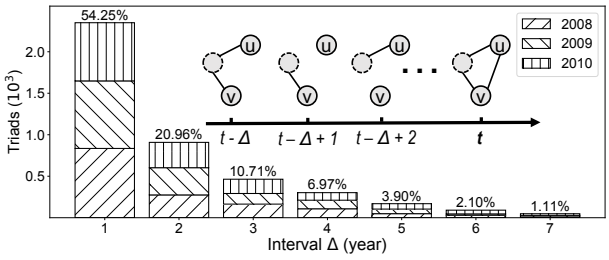
Most graph neural network models learn embeddings of nodes in *static* attributed graphs for predictive analysis. Recent attempts have been made to learn temporal proximity of the nodes. We find that real dynamic attributed graphs exhibit complex *co-evolution* of node attributes and graph structure. Learning node embeddings for forecasting *change of node attributes* and *birth and death of links* over time remains an open problem. In this work, we present a novel framework called CoEvoGNN for modeling dynamic attributed graph sequence. It preserves the impact of earlier graphs on the current graph by embedding generation through the sequence. It has a temporal self-attention mechanism to model long-range dependencies in the evolution. Moreover, CoEvoGNN optimizes model parameters jointly on two dynamic tasks, attribute inference and link prediction over time. So the model can capture the co-evolutionary patterns of attribute change and link formation. This framework can adapt to any graph neural algorithms so we implemented and investigated three methods based on it: CoEvoGCN, CoEvoGAT, and CoEvoSAGE. Experiments demonstrate the framework (and its methods) outperform strong baselines on predicting an entire unseen graph snapshot of personal attributes and interpersonal links in dynamic social graphs and financial graphs.

1 INTRODUCTION

Graphs are ubiquitous in the world and real graphs evolve over time via individual behaviors. For example, social network users establish and/or remove links between each other via the behaviors of following, mentioning, replying, and etc. The user's attributes such as textual features from generated content are also changing. These two types of dynamics, social links and user attributes, have impact on each other. Specifically, on academic co-authorship networks, researchers are looking for collaborators (reflected as neighbor nodes) who have similar or complementary knowledge [26] (which may be reflected as published keywords, a type of node attributes). And their personal research topics may change according to new collaborations. The co-evolutionary patterns of node attributes and graph structure are complex yet valuable, and need to be effectively learned for forecasting future attributes and structures in graph-based applications.



(a) If links at time t appeared previously, more than 29% were at least two steps earlier ($\Delta \geq 2$).



(b) If links at time t could be created by closing a triad in previous graphs, more than 45% were at least two time steps earlier ($\Delta \geq 2$).

Figure 1: The formation of a new link in co-authorship networks depends on more than one previous graphs.

Graph Neural Networks (GNNs) have been widely studied for learning representations of nodes from static graph data for various tasks such as node classification [11], community detection [3], and link prediction [7]. There have been dynamic graph learning methods that explore the idea of combining GNN with recurrent neural network (RNN) for dynamic attributed graphs. WD-GCN [16] stacked an LSTM [8] on top of a GCN [11] module and CD-GCN [16] added a skip connection above it. GCRN [21] explored a similar architecture and proposed a modified LSTM by replacing fully connected layers with graph convolution layers [6]. However, these pioneering methods still relied on a fair amount of information in current graphs (though which can be incomplete) and thus were not capable of forecasting an entire snapshot of attributed graph.

Recently, EvolveGCN [18] was proposed to address this issue using GRU [4] to learn the parameter changes in GCN [11] instead of node embedding changes. Specifically, the GCN's weight matrices were treated as hidden states and node embeddings were fed into the GRU at each time. This method iteratively generated node embeddings and, in turn, injected temporal information into the GCN model. However, it has three limitations. First, like other RNN-based

DLG '20, August 23–27, 2020, Virtual Event, CA, USA

ACM ISBN 978-1-4503-9999-9/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

methods, it has inherent difficulty in compressing long-range dependencies into hidden states [2] as well as severe scalability issues as they cannot be parallelized [23]. The time complexity is largely intractable: the number of times of applying the GRU module grows proportionally with the number of nodes in the data. Second, it assumes the underlying force driving the graph evolution only comes from the changes of links. It is unaware of the co-evolutionary process between node attributes and graph structure. Third, its design is specific to the choice of the GCN algorithm. When different graph neural algorithms (e.g., GCN, GAT [24], GraphSAGE [7]) have different advantages and deliver data-dependent performances, we expect to apply the dynamic method upon all the algorithms; however, it is unclear how to build EvolveGCN upon any other algorithm that is parameterized by more than one matrix layer-wise.

In this work, we propose a novel framework Co-Evolutionary Graph Neural Networks (CoEvoGNN). First, we design an S-stack temporal self attention architecture as the core component of CoEvoGNN. It learns the impact of multiple previous graph snapshots on the current one with self-adapting importance so that it can effectively capture the *evolutionary* patterns in graph sequence. Its temporal self-attention mechanism makes the time complexity grow linearly with the increase of training range. And it remains fully parallelizable compared to existing RNN-based methods. Second, we devise a multi-task loss function that optimizes CoEvoGNN jointly on predicting node attributes and graph structure over time. This allows our framework to learn the *co-evolutionary* interactions between change of attributes and formation of links, and to use these valuable information to better forecast an unseen future graph snapshot. Besides, our framework can utilize any static graph neural algorithm for aggregating neighbor information along the structural axis. We developed and investigated three (but not limited to three) methods based on the proposed framework, named CoEvoGCN, CoEvoGAT, and CoEvoSAGE. We evaluate the performance of CoEvoGNNs methods on forecasting an entire future snapshot of co-authorship attributed graph and virtual currency graph. Experimental results demonstrate it can outperform competitive baselines by +9.2% of F1 score on link prediction, and by −49.1% of RMSE on attribute inference.

2 THE CO-EVOLUTION PHENOMENON

The co-evolutionary process of node attributes and graph structure in real dynamic graphs is a fundamentally complex phenomenon and imposes great challenges for learning. *First, the node attributes and structure of a graph snapshot depend on the states of multiple previous graphs with an effect of time decay* [14]. Take a co-authorship network as an example: the formation of a collaboration link between two authors can be traced back to their previous co-authored event 2, or 3, or even 5 years ago. In Figure 1(a), we plot the distribution of two author nodes developing a future link at $t \in \{2008, 2009, 2010\}$ if they were linked at $t - \Delta$. The proportion of these links are presented by the minimum interval Δ. Though a fair amount of the links occurred in the last year ($\Delta = 1$), around 29% of new links can be traced back to previous years of $\Delta > 1$. In Figure 1(b), we plot another important mechanism of link formation – triad closure [5]. It is evident that 46% links formed through this process fell in the range of $\Delta > 1$, though the number quickly drops

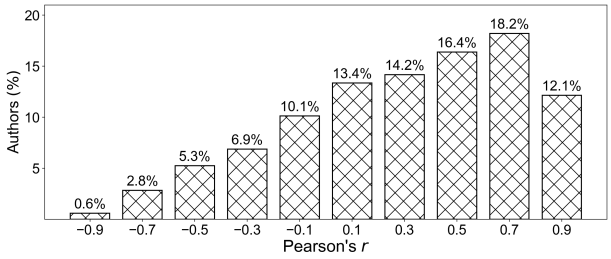


Figure 2: The evolution of personal attributes (i.e., keyword change) and the evolution of graph structure (i.e., collaborator change) are highly correlated.

at longer intervals. This indicates that earlier graph states contain valuable information for predicting the future, and their relative importance should be fully considered.

Second, node attributes and graph structure mutually influence each other. In a co-authorship network, forming a new link (i.e., a new collaboration) extends research scope and increases the impact of authors. And, having new research topics, or a higher h-index, in turn helps the author to develop new collaborations [27]. Figure 2 shows the distribution of Pearson correlation between attribute and link evolutions. For every author, we calculate the Jaccard similarity of keyword sets and that of collaborator sets between two years. Then we measure the correlation between the two similarity series over time. If an author changed his/her keywords significantly and his/her collaborators also changed significantly, the correlation would be high. We spot that more than 60% of the authors show higher-than-0.3 correlation. This mutually influencing characteristic between node attributes and graph structure requires both types of information to be used for training the model. Existing methods were not able to learn effective node embeddings for simultaneously forecasting node attributes and graph structure.

3 PROBLEM DEFINITION

Traditionally, a static graph is represented as $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ denotes the set of nodes and $\mathcal{E}$ denotes the set of edges. The node attribute matrix of G is denoted as $\mathbf{X} \in \mathbb{R}^{n \times r}$, where each row $\mathbf{x}_v$ describes the r-dimensional raw attribute vector of node v. However, real graphs evolve over time. The evolutionary process manifests in two aspects: (1) the change of node attributes $\mathbf{X}^t$ across time steps $t = 0, 1, \ldots, T$; and, (2) the change of graph structure $G^t = (\mathcal{V}, \mathcal{E}^t)$ across time. For brevity, we use $\mathcal{V}$ to denote all unique nodes, i.e., $\mathcal{V} = \bigcup_{t=0}^{T} \mathcal{V}^t$, so the change in G^t is reflected as the change of $\mathcal{E}^t$. We define a sequence of dynamic graphs as:

Definition 3.1 (Dynamic Graph Sequence). A dynamic graph sequence across time steps from 0 to T contains consecutive snapshots $(G^0, \mathbf{X}^0), (G^1, \mathbf{X}^1), ..., (G^T, \mathbf{X}^T)$ of both the graph structure and node attributes. Each single snapshot $(G^t, \mathbf{X}^t)$ for $t = 0, 1, \ldots, T$ represents a transitional state of the graph during the evolution.

Then, we formally define the research problem as follows:

Problem: *Given* a dynamic graph sequence $\mathcal{D} = \{(G^t, \mathbf{X}^t) \mid t = 0, 1, \ldots, T\}$, *learn* a mapping function $f(\mathcal{D}) : \mathcal{V} \times \{0, 1, \ldots, T\} \rightarrow \mathbb{R}^d$ that embeds each node $v \in \mathcal{V}$ into a d-dimensional (typically $d \ll r, |\mathcal{V}|$) representation vector $\mathbf{h}_v^t$ at each time step t that can preserve co-evolution of node attributes and graph structure.

For a non-trivial dynamic graph sequence with $T \geq 1$, each $\mathbf{H}^t$ should contain information not only about the current snapshot $(G^t, \mathbf{X}^t)$, but also summarize the co-evolution trend from recent past into near future. Specifically, we aim at learning $\mathbf{H}^t$ that can be characterized by the following two properties:

- Revealing the historical co-evolution trend information of node attributes and graph structure in previous S graph snapshots $(G^{t-S}, \mathbf{X}^{t-S}), \ldots, (G^{t-1}, \mathbf{X}^{t-1})$.
- Being highly indicative about the developing co-evolution of node attributes and graph structure of next S graph snapshots in future $(G^{t+1}, \mathbf{X}^{t+1}), \ldots, (G^{t+S}, \mathbf{X}^{t+S})$.

4 PROPOSED FRAMEWORK

In this section, we present the evolutionary node embedding generation process of CoEvoGNN as illustrated in Figure 3 (a). The pseudocode of our proposed framework is given in Algorithm 1. CoEvoGNN is designed to capture the co-evolution pattern of node attributes and graph structure in dynamic graph sequence along the temporal axis.

Given a dynamic graph sequence $\{(G^t, \mathbf{X}^t) \mid t = 0, \ldots, T\}$, CoEvoGNN's weight matrices $\{\mathbf{W}^{\langle s \rangle} \mid s = 1, \ldots, S\}$ and its fusion matrix $\mathbf{\Gamma}$, the temporal evolution span S, and a set of static models $\{f_{static}^{\langle s \rangle} \mid s = 1, \ldots, S\}$, CoEvoGNN first generates the initial latent embedding of node from the leading graph snapshot $(G^0, \mathbf{X}^0)$ (Line 3 of Algo. 1). In practice, we can use an arbitrary static GNN algorithm (e.g., GCN [11], GAT [24] and GraphSAGE [7]) as $f_{static}^{\langle \cdot \rangle}$ functions. We will examine the choice of $f_{static}^{\langle \cdot \rangle}$ in Section 5. In particular, we concatenate the intermediate node embeddings at different structural depths together, i.e., $\mathbf{h}_v^{\langle \cdot \rangle} = f_{static}^{\langle \cdot \rangle}(v \mid (G, \mathbf{X}), L) \in \mathbb{R}^{dL \times 1}$, where d is the latent dimensions and L is the structural depth. This can allow CoEvoGNN to retain complete high-order neighbor structural information from $f_{static}^{\langle \cdot \rangle}$ across time [22, 25], and later determine the relative importance of previous graphs.

After initialization, CoEvoGNN generates latent node embeddings along time steps $t = 1, \ldots, T$ in a cascade mode. For node v at a specific time step t, CoEvoGNN extracts and merges its neighbor structural embeddings in the last S, or precisely $\min(t, S)$, snapshots with self-adapting importance (Line 4-19 in Algo. 1). The newly fused $\mathbf{h}_v^t$ gets l2 normalized and returned as the output node latent embedding (Line 20 in Algo. 1). Next, we introduce the design of CoEvoGNN's core component for automatically distilling and fusing influence from multiple previous graph snapshots.

4.1 S-stack temporal self-attention

Equipping with static graph neural methods $f_{static}^{\langle \cdot \rangle}$ as its underlying aggregator, CoEvoGNN is able to distill structural information from each single time step independently. This means the resulting node embeddings $\mathbf{H}^t$ are solely determined by its corresponding graph snapshot $(G^t, \mathbf{X}^t)$, and all evolutionary dynamics of the graph are ignored. How can we effectively capture the co-evolution of node attributes and graph structure along the temporal axis? One straightforward way is to enforce the Markov property [1] and directly transform node embeddings from the previous time step $\mathbf{H}^{t-1}$ into the current one $\mathbf{H}^t$ [19]. But this oversimplified setting does not always hold in real cases. As an example: in an evolutionary co-authorship graph, authors collaborate in one year does not necessarily indicate they will collaborate in the next year; but authors could be more likely to collaborate if they have collaboration experience before [10]. Alternatively, we could assume node embeddings at each time $\mathbf{H}^t$ depend on all previous node embeddings $\mathbf{H}^0, \cdots, \mathbf{H}^{t-1}$, following a strict autoregressive paradigm [13]. Most related methods fall in this category and utilizes various RNN models to capture the dynamics of node embeddings [16, 21] or GNN parameters [18]. However, these models have difficulty in compressing long-range dependencies into hidden state [2], as well as severe scalability issues as they cannot be easily parallelized [23].

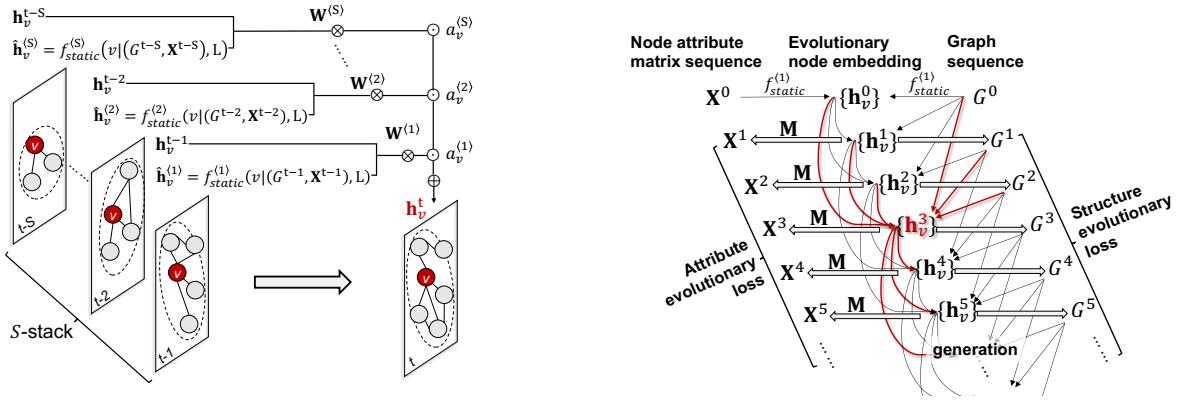
To this end, we design a novel *S-stack temporal self-attention* architecture (see Figure 3 (a)) for automatically distilling and fusing influence from multiple previous graph snapshots. Particularly, for node v at time step t, we first leverage static models $f_{static}^{\langle \cdot \rangle}$ to obtain its rich neighbor structural information $\hat{\mathbf{h}}_v^{\langle \cdot \rangle}$ (where $\langle \cdot \rangle$ indicates the temporal depth from the previous snapshot to the current one) for each one of the last S, or precisely $\min(t, S)$, snapshots (Line 4-8 of Algo. 1). Each one of these neighbor structural information embeddings $\hat{\mathbf{h}}_v^{\langle s \rangle} \in \mathbb{R}^{dL \times 1}$ is also processed into the pre-attention

Algorithm 1: CoEvoGNN framework

Input : Dynamic graph sequence $\{(G^t, \mathbf{X}^t) \mid t = 0, \ldots, T\}$; parameter matrices $\{\mathbf{W}^{\langle s \rangle} \mid s = 1, \ldots, S\}$ and fusion matrix $\mathbf{\Gamma}$; temporal evolution span S; and, static graph neural models $\{f_{static}^{\langle s \rangle} \mid s = 1, \ldots, S\}$.

Output : Node latent embeddings $\mathbf{h}_v^t$, $v \in \mathcal{V}$ and $1 \leq t \leq T$.

1 **for** $v \in \mathcal{V}$ **do**
2 // Initialization
3 $\mathbf{h}_v^0 \leftarrow f_{static}^{\langle 1 \rangle}(v \mid (G^0, \mathbf{X}^0), 1)$
4 **for** $t = 1, \ldots, T$ **do**
5 // Structural aggregations
6 Let $\hat{H}_v[1, \ldots, \min(t, S)]$ and $E_v[1, \ldots, \min(t, S)]$ be new arrays
7 **for** $s = 1, \ldots, \min(t, S)$ **do**
8 $\hat{\mathbf{h}}_v^{\langle s \rangle} \leftarrow f_{static}^{\langle s \rangle}(v \mid (G^{t-s}, \mathbf{X}^{t-s}), L)$
9 $e_v^{\langle s \rangle} \leftarrow (\mathbf{h}_v^{t-s})^\top \cdot \mathbf{\Gamma} \cdot \hat{\mathbf{h}}_v^{\langle s \rangle}$
10 $\hat{H}_v[s] = \hat{\mathbf{h}}_v^{\langle s \rangle}$ and $E_v[s] = e_v^{\langle s \rangle}$
11 **end**
12 // Temporal self-attention
13 Let $A_v[1, \ldots, \min(t, S)]$ be a new array
14 **for** $s = 1, \ldots, \min(t, S)$ **do**
15 $a_v^{\langle s \rangle} \leftarrow \frac{\exp(E_v[s])}{\sum_{s'=1}^{\min(t,S)} \exp(E_v[s'])}$
16 $A_v[s] = a_v^{\langle s \rangle}$
17 **end**
18 // Fusion and normalization
19 $\mathbf{h}_v^t \leftarrow \sum_{s=1}^{\min(t,S)} A_v[s]\, \sigma\left(\mathbf{W}^{\langle s \rangle} \cdot \left[\mathbf{h}_v^{t-s}; \hat{H}_v[s]\right]\right)$
20 $\mathbf{h}_v^t \leftarrow \mathbf{h}_v^t / \left\|\mathbf{h}_v^t\right\|_2$
21 **end**
22 **end**



(a) Evolutionary node embedding generation ($t \geq S$)

(b) Evolutionary attribute and structure losses ($S = 3$)

Figure 3: Visual illustration of CoEvoGNN's evolutionary embedding generation and co-evolutionary loss function

energy scalar $e_v^{\langle s \rangle}$ by feeding it into a bilinear mapping $\mathbf{\Gamma} \in \mathbb{R}^{d \times dL}$ along with the node latent embedding at the same time step $\mathbf{h}_v^{t-s} \in \mathbb{R}^{d \times 1}$ (Line 9 of Algo. 1). Next, node v's self-adapting weights $a_v^{\langle \cdot \rangle}$ for fusing previous influence are calculated from $e_v^{\langle \cdot \rangle}$ by taking softmax over them (Line 12-17 of Algo. 1). Then, for each one of the previous S-stack, the node latent embedding $\mathbf{h}_v^{t-s}$ and its neighbor structural embedding $\hat{H}_v[s] = \hat{\mathbf{h}}_v^{\langle s \rangle}$ are concatenated and transformed the through the weight matrices $\mathbf{W}^{\langle s \rangle} \in \mathbb{R}^{d \times (d+dL)}$ (Line 19 of Algo. 1). At last, the new node embedding $\mathbf{h}_v^t \in \mathbb{R}^{d \times 1}$ with self-attention on transformed previous latent and structural embeddings according to $A_v = a_v^{\langle 1 \rangle}, \ldots, a_v^{\langle \min(t,S) \rangle}$ are returned.

At a high level, CoEvoGNN merges each node's latent embeddings and neighbor structural information embeddings for up to S previous time steps. This is different from solely relying on the most recent time step or compressing information from all previous time steps which can easily leads to unaffordable efficiency. On one hand, the temporal evolution span hyperparameter S controls a tradeoff between the model's expressive power of co-evolution pattern and its space complexity; on the other hand, it allows the adaptability for handling specific data or applications as increasing S brings diminishing marginal benefits in practice. Furthermore, the temporal self-attention mechanism on S-stack grants each node the flexibility for judging the relative importance of previous graphs and dynamically fusing them into the current node latent embedding.

4.1.1 Inferring future node embeddings. The output of CoEvoGNN consists of a sequence of node latent embeddings $\mathbf{H}^t$, $t = 1, \cdots, T$, summarizing the training dynamic graph sequence. At inference phase, beyond the training range, CoEvoGNN generates an arbitrary number of node latent embeddings at future time steps (e.g., $\mathbf{H}^{T+1}, \mathbf{H}^{T+2}, \ldots$). The future node embeddings directly reflect CoEvoGNN's forecasting capability on the co-evolutoin trend of node attributes and graph structure learned from the observed graph snapshots. Forecasting into far future would be really challenging. In this paper, we only focus on predicting node embeddings of the next time step $(T + 1)$ after the training evolutionary graph snapshots and leave forecasting multiple time steps as future work. Next, we introduce the training procedure and objective of CoEvoGNN.

4.2 Training on multi-task co-evolutionary loss

In this section, we present the training process of CoEvoGNN. The overall loss function is defined in Eqn. (1) and the training procedure of CoEvoGNN is presented in Algorithm 2.

To learn the CoEvoGNN model on a dynamic graph sequence for forecasting into future, we carefully devise a multi-task loss function supervising generated node latent representations $\mathbf{h}_v^t$ over training time steps $t = 1, \ldots, T$. In a forward pass, for each mini-batch of nodes $\mathcal{V}' \subset \mathcal{V}$, the result embeddings gets evaluated by the overall loss. During backpropagation, we use stochastic gradient descent to update the set of weight matrices $\{\mathbf{W}^{\langle s \rangle} \mid s = 1, \ldots, S\}$, the fusion matrix $\mathbf{\Gamma}$, and attribute transformation matrix $\mathbf{M}$ (see Section 4.2.1), which parameterizes the proposed CoEvoGNN model.

$$\min_{\mathbf{h}_v^t, v \in \mathcal{V}', t=1,\ldots,T} \mathcal{J} = \sum_{t=1}^{T} \sum_{v \in \mathcal{V}'} \alpha \mathcal{J}_{\mathbf{X}^t}(\mathbf{h}_v^t) + (1-\alpha) \mathcal{J}_{G^t}(\mathbf{h}_v^t). \quad (1)$$

This multi-task evolutionary objective is mainly composed of two terms: the attribute evolutionary loss $\mathcal{J}_{\mathbf{X}^t}$, and the structure evolutionary loss $\mathcal{J}_{G^t}$. A mixture hyperparameter α is used to balance the magnitude of these two terms.

4.2.1 Attribute evolutionary loss for attribute inference. The attribute evolutionary loss $\mathcal{J}_{\mathbf{X}^t}$ is defined as below:

$$\mathcal{J}_{\mathbf{X}^t}(\mathbf{h}_v^t) = \|\sigma(\mathbf{M} \cdot \mathbf{h}_v^t) - \mathbf{x}_v^t\|_F^2, \quad (2)$$

where $\mathbf{M}$ is the attribute transformation matrix and σ is non-linear function such as ReLU or sigmoid. Given a node latent embedding $\mathbf{h}_v^t \in \mathbb{R}^{d \times 1}$, the attribute transformation matrix $\mathbf{M} \in \mathbb{R}^{r \times d}$ is used for mapping $\mathbf{h}_v^t$ back into the r-dim raw attribute space. Node v's remapped attribute inference vector $\sigma(\mathbf{M} \cdot \mathbf{h}_v^t) \in \mathbb{R}^{r \times 1}$ is then compared against the true node attribute vector $\mathbf{x}_v^t$ by measuring the L2 distance. Note that parameter matrix $\mathbf{M}$, which is irrelevant to T or S, describes the transformation from latent space back to raw attribute space, also gets updated with back propagation.

4.2.2 Structure evolutionary loss for link prediction. The structure evolutionary loss $\mathcal{J}_{G^t}$ is defined as below:

$$\begin{aligned} \mathcal{J}_{G^t}(\mathbf{h}_v^t) \quad = \quad & -\log\left(\sigma\left((\mathbf{h}_v^t)^\top \cdot \mathbf{h}_u^t\right)\right) \\ & -Q \cdot \mathbb{E}_{u' \sim P_n(v)} \log\left(\sigma\left(-(\mathbf{h}_v^t)^\top \cdot \mathbf{h}_{u'}^t\right)\right), \end{aligned} \quad (3)$$

Table 1: On co-authorship attributed graph sequence, CoEvoGNN outperforms baselines on forecasting $\mathbf{X}^{2010}$ and G^{2010}.

	$\mathcal{D}_{\text{AU}}^{2K}$					$\mathcal{D}_{\text{AU}}^{10K}$				
	Attributes $\mathbf{X}^{2010}$		Links in G^{2010}			Attributes $\mathbf{X}^{2010}$		Links in G^{2010}		
Method	MAE	RMSE	AUC	F1	P@50, 100, 200	MAE	RMSE	AUC	F1	P@50, 100, 200
GCN [11]	0.649	1.297	0.082	0.196	0.34, 0.42, 0.36	0.742	1.566	0.034	0.071	0.30, 0.40, 0.34
GAT [24]	0.658	1.342	0.075	0.192	0.34, 0.36, 0.36	0.758	1.628	0.028	0.053	0.32, 0.30, 0.32
GraphSAGE [7]	0.643	1.265	0.084	0.201	0.38, 0.44, 0.41	0.729	1.438	0.039	0.078	0.36, 0.40, 0.42
DynamicTriad [30]	N/A	N/A	0.112	0.241	0.76, 0.62, 0.60	N/A	N/A	0.058	0.147	0.60, 0.59, 0.57
DySAT [20]	N/A	N/A	0.120	0.222	0.54, 0.46, 0.38	N/A	N/A	0.036	0.127	0.48, 0.43, 0.36
DCRNN [15]	0.458	0.960	0.019	0.073	0.12, 0.10, 0.10	0.423	0.853	0.006	0.027	0.09, 0.06, 0.03
STGCN [29]	0.478	1.127	0.006	0.027	0.04, 0.02, 0.04	0.567	1.589	0.001	0.007	0.04, 0.04, 0.02
EvolveGCN [18]	0.684	1.279	0.133	0.256	0.78, **0.80**, 0.67	0.768	1.603	0.069	0.161	0.69, 0.74, **0.59**
CoEvoGCN	0.452	0.944	0.147	0.269	**0.82**, 0.76, 0.69	0.414	0.831	0.076	0.167	0.78, **0.76**, 0.54
CoEvoGAT	0.453	0.946	0.143	0.271	0.78, 0.74, 0.66	0.415	0.831	0.075	0.167	0.78, **0.76**, 0.54
CoEvoSAGE	**0.449**	**0.938**	**0.151**	**0.274**	**0.82**, **0.80**, **0.72**	**0.410**	**0.828**	**0.079**	**0.170**	**0.80**, **0.76**, 0.58

Algorithm 2: Training procedure of CoEvoGNN

Initialize model parameters $\{\mathbf{W}^{\langle s \rangle} \mid s = 1, \ldots, S\}$, $\mathbf{\Gamma}$, and $\mathbf{M}$
repeat
  Sample minibatch of nodes $\mathcal{V}'$ from all nodes $\mathcal{V}$
  $\mathbf{H}^1, \ldots, \mathbf{H}^T \leftarrow$ CoEvoGNN($\mathcal{V}'$) ▷ see Algorithm 1
  // Compute evolutionary losses
  $\mathcal{J}_{\mathbf{X}^1}, \ldots, \mathcal{J}_{\mathbf{X}^T} \leftarrow$ Compute the attribute evolutionary loss for attribute inference ▷ see Equation (2)
  $\mathcal{J}_{G^1}, \ldots, \mathcal{J}_{G^T} \leftarrow$ Compute the structure evolutionary loss for link prediction ▷ see Equation (3)
  $\mathcal{J} \leftarrow$ Compute overall loss ▷ see Equation (1)
  // Update parameters
  $\mathbf{W}^{\langle \cdot \rangle} \overset{+}{\leftarrow} -\nabla_{\mathbf{W}^{\langle \cdot \rangle}}(\mathcal{J})$
  $\mathbf{\Gamma} \overset{+}{\leftarrow} -\nabla_{\mathbf{\Gamma}}(\mathcal{J})$
  $\mathbf{M} \overset{+}{\leftarrow} -\nabla_{\mathbf{M}}(\mathcal{J})$
until *finish*;

where node u is one of the 1st-order neighbors of node v. This can be relaxed to that node u co-occurs near node v on a fixed-length random walk. Node u' is a negative sample node, i.e., disconnected node with v, drawn according to the negative sampling distribution $P_n(v)$. Q is the number of negative samples and σ is the non-linear function. Intuitively, Eqn. (3) pulls similar nodes closer and pushes dissimilar nodes away in the latent space. Taken together with Eqn. (2), the multi-task evolutionary loss function (Eqn. (1)) captures the co-evolution of node attributes and graph structure over time.

4.3 Complexity Analysis

Assuming the per-batch time complexity of CoEvoGNN's underlying static methods $f_{static}^{\langle \cdot \rangle}$ is $O\left(\Pi_{l=1}^{L} s_l\right)$ in principle [7] (where L is the structural depth and s_l is the neighbor sampling size at the l-th layer) and they can be parallelized in the S-stack temporal self-attention architecture, the CoEvoGNN's per-batch time complexity is $O\left(T \Pi_{l=1}^{L} s_l\right)$. The computation cost only increases linearly with training range T and is regardless of the temporal evolution span S.

5 EXPERIMENTS

In this section, we evaluate CoEvoGNN on two forecasting tasks: (1) node attribute prediction, and (2) graph link prediction. In all experiments, we test on predicting the *next graph snapshot*.

5.1 Datasets

We used 4 datasets from two type of evolutionary graphs.

Evolutionary co-authorship graph. We built a sequence of yearly co-authorship graphs by collecting 226, 611 papers from 2001 to 2010 in computer science from Microsoft Academic Graph [28]. Authors were ranked by their number of papers. The top 2, 000 and 10, 000 were used to make two datasets denoted by $\mathcal{D}_{\text{AU}}^{2K}$ and $\mathcal{D}_{\text{AU}}^{10K}$. The venues and the paper title's words were used as node attributes after filtering out infrequent ones. As a result, we have 316 venues and 3, 549 words in $\mathcal{D}_{\text{AU}}^{2K}$; and 448 venues, 6, 442 words in $\mathcal{D}_{\text{AU}}^{10K}$.

Evolutionary virtual currency graph. We used 2 benchmark datasets Bitcoin-OTC and Bitcoin-Alpha of Bitcoin transaction networks [12] denoted by $\mathcal{D}_{\text{BC}}^{\text{otc}}$ and $\mathcal{D}_{\text{BC}}^{\text{alp}}$. We followed the treatments as in [18] to form a sequence of graphs with 138 time steps (each for about 2 weeks), and use node in/out degree as input features.

5.2 Experimental settings

Baseline methods: We compare CoEvoGNN's variants using representative static methods against dynamic graph neural methods:

- GCN [11], GAT [24] and GraphSAGE [7]: We incorporate each one of these static method as CoEvoGNN's underlying operation and denote them as CoEvoGCN, CoEvoGAT, and CoEvoSAGE, respectively. We also directly compare against these static methods taking merged graphs as input.
- DynamicTriad [30] and DySAT [20]: These two methods cannot handle node attributes. All graphs are fed for training, and we focus on the task of future graph link prediction.
- DCRNN [15] and STGCN [29]: The most recent graph and all node attributes are used for training. The final prediction matrix is used for the future node attribute prediction. And, the node embeddings outputted by the diffusion convolutional layer of DCRNN, or the spatio-temporal convolutional block of STGCN are used for future graph link prediction.

- EvolveGCN [18]: All graph snapshots are provided as input. We use its link prediction loss for training, and use node embeddings outputted by the last evolving graph convolution unit for future graph link and node attributes prediction.

We use open-source implementations provided by the original paper for all baseline methods and follow the recommended setup guidelines when possible. **Evaluation metrics:** For node attribute prediction, we use Mean Average Error (MAE) and Root Mean Squared Error (RMSE); for link prediction, we use Area Under the precision-recall Curve (AUC), F1 measure, and Precision@50, 100, 200.

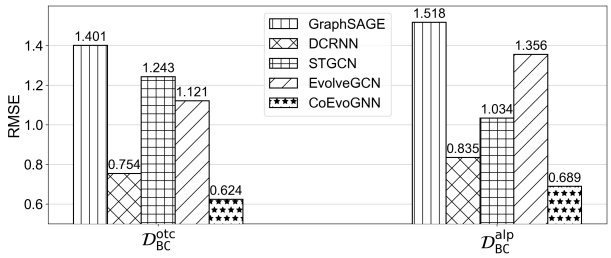
5.3 Performance

Table 1 presents results on co-authorship graphs $\mathcal{D}_{\text{AU}}^{2K}$ and $\mathcal{D}_{\text{AU}}^{10K}$. We report the performance of static methods GCN, GAT and GraphSAGE trained using all historical graph snapshots. But simply merging and feeding all previous graph snapshots into a static model loses the co-evolutionary patterns and thus underperforms almost all dynamic methods. It verifies that static methods cannot accurately forecast node attributes and graph structure. Three variants of CoEvoGNNs perform similar to each other; CoEvoSAGE makes slightly lower RMSE values and higher F1 values on both datasets. Without causing ambiguity, we refer CoEvoSAGE as CoEvoGNN for comparison in this section. Figure 4 presents the results on evolutionary virtual currency graphs $\mathcal{D}_{\text{BC}}^{\text{otc}}$ and $\mathcal{D}_{\text{BC}}^{\text{alp}}$.

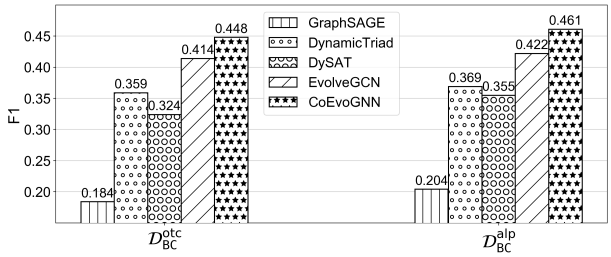
Both dynamic network embedding methods DynamicTriad and DySAT give comparable performance to CoEvoGNN on the task of future graph link prediction. However, they only consider the dynamics of evolving graph structure instead of capturing the co-evolution of node attributes and graph structure. In contrast, by fusing influence from multiple previous states, CoEvoGNN can give higher F1 scores compared with DynamicTriad. This tells considering node attribute evolution is beneficial for modeling the change of graph structure as they are mutually influencing each other. They should be jointly modeled as a co-evolutionary pattern.

For spatiotemporal forecasting methods DCRNN and STGCN, they are designed for modeling the change of node attributes assuming the graph structure remains static. DCRNN outperforms all other baseline methods on the task of future node attribute prediction, but it cannot produce acceptable performance on the task of future graph link prediction. The proposed CoEvoGNN is able to score lower RMSEs compared with DCRNN; and, at the same time, perform much better on the task of future graph link prediction. This again demonstrates the advantage of CoEvoGNN by modeling the co-evolutionary pattern of node attributes and graph structure as they are mutually influencing each other.

The most competitive baseline EvolveGCN achieves the best performance for predicting future graph links among all others. Although its input also includes all historical graph snapshots, one fundamental difference between EvolveGCN and our CoEvoGNN is that EvolveGCN assumes the underlying force driving the graph evolution only comes from the changes in graph structure. It can be trained under its node classification mode but that requires the class information for each node at each time step which is commonly unavailable. In either way, EvolveGCN is unaware of the co-evolution process between node attributes and graph structure. So, EvolveGCN can only generate future node attribute predictions of similar quality as the static model GraphSAGE.



(a) Models' performance on the task of future node attribute prediction. Lower RMSE bar is better. (DynamicTriad and DySAT not applicable)



(b) Models' performance on the task of future graph link prediction. Higher F1 bar is better. (DCRNN and STGCN excluded)

Figure 4: CoEvoGNN outperforms baseline methods on forecasting an entire future snapshot of virtual currency graph.

6 RELATED WORK

CTDNE [17] proposed to model temporal structure dependencies in continuous-time dynamic networks by conducting temporal random walks. DynamicTriad [30] preserved the dynamic structural information by modeling the triadic closure process in network. DySAT [20] employed a self-attention mechanism over both neighbor nodes and historical representations. These methods were not designed to handle node attributes. They can neither capture the evolution pattern of node attributes nor forecast future attribute information. DCRNN [15] modeled the traffic flow as a diffusion process on a directed graph and adopted an encoder-decoder architecture for capturing the temporal attribute dependencies. STGCN [29] modeled the traffic network as a general graph and employed a fully convolutional structure [6] on the temporal axis. These methods assume the graph structure remains static all the time, thus being incapable of capturing the evolution of graph structure or forecasting into future graph structure [9].

7 CONCLUSIONS

In this work, we proposed a new framework for learning node embeddings from evolutionary attributed graph and inferring future node representations. It aggregated the information in previous snapshots to the current one using temporal self-attention and employed a multi-task loss function based on attribute inference and link prediction over time. Experimental results demonstrated our method outperformed strong baselines on forecasting an entire future snapshot of co-authorship and virtual currency network.

ACKNOWLEDGMENTS

This work was supported in part by NSF Grant IIS-1849816.

REFERENCES

[1] Charu Aggarwal and Karthik Subbian. 2014. Evolutionary network analysis: A survey. *ACM Computing Surveys (CSUR)* 47, 1 (2014), 1–36.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[3] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2013. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203* (2013).

[4] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[5] James S Coleman and James Samuel Coleman. 1994. *Foundations of social theory*. Harvard university press.

[6] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *NeurIPS*. 3844–3852.

[7] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in neural information processing systems*. 1024–1034.

[8] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[9] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous graph transformer. In *Proceedings of The Web Conference 2020*. 2704–2710.

[10] Meng Jiang, Christos Faloutsos, and Jiawei Han. 2016. Catchtartan: Representing and summarizing dynamic multicontextual behaviors. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 945–954.

[11] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[12] Srijan Kumar, Bryan Hooi, Disha Makhija, Mohit Kumar, Christos Faloutsos, and VS Subrahmanian. 2018. Rev2: Fraudulent user prediction in rating platforms. In *WSDM*. 333–341.

[13] Hugo Larochelle and Iain Murray. 2011. The neural autoregressive distribution estimator. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. 29–37.

[14] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2007. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 1, 1 (2007), 2.

[15] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2017. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926* (2017).

[16] Franco Manessi, Alessandro Rozza, and Mario Manzo. 2020. Dynamic graph convolutional networks. *Pattern Recognition* 97 (2020), 107000.

[17] Giang Hoang Nguyen, John Boaz Lee, Ryan A Rossi, Nesreen K Ahmed, Eunyee Koh, and Sungchul Kim. 2018. Continuous-time dynamic network embeddings. In *WWW*. 969–976.

[18] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, and Charles E Leisersen. 2020. Evolvegcn: Evolving graph convolutional networks for dynamic graphs. (2020).

[19] Meng Qu, Yoshua Bengio, and Jian Tang. 2019. Gmnn: Graph markov neural networks. *arXiv preprint arXiv:1905.06214* (2019).

[20] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, and Hao Yang. 2018. Dynamic Graph Representation Learning via Self-Attention Networks. *arXiv preprint arXiv:1812.09430* (2018).

[21] Youngjoo Seo, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. 2018. Structured sequence modeling with graph convolutional recurrent networks. In *ICONIP*. 362–373.

[22] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *CVPR*. 2818–2826.

[23] 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 *NeurIPS*. 5998–6008.

[24] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[25] Daheng Wang, Meng Jiang, Munira Syed, Oliver Conway, Vishal Juneja, Sriram Subramanian, and Nitesh V Chawla. 2020. Calendar Graph Neural Networks for Modeling Time Structures in Spatiotemporal User Behaviors. *arXiv preprint arXiv:2006.06820* (2020).

[26] Daheng Wang, Meng Jiang, Qingkai Zeng, Zachary Eberhart, and Nitesh V Chawla. 2018. Multi-type itemset embedding for learning behavior success. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2397–2406.

[27] Daheng Wang, Tianwen Jiang, Nitesh V Chawla, and Meng Jiang. 2019. Tube: Embedding behavior outcomes for predicting success. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1682–1690.

[28] Kuansan Wang, Zhihong Shen, Chiyuan Huang, Chieh-Han Wu, Yuxiao Dong, and Anshul Kanakia. 2020. Microsoft Academic Graph: When experts are not enough. *Quantitative Science Studies* 1, 1 (2020), 396–413.

[29] Bing Yu, Haoteng Yin, and Zhanxing Zhu. 2017. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *arXiv preprint arXiv:1709.04875* (2017).

[30] Lekui Zhou, Yang Yang, Xiang Ren, Fei Wu, and Yueting Zhuang. 2018. Dynamic network embedding by modeling triadic closure process. In *Thirty-Second AAAI Conference on Artificial Intelligence*.