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.

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
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 networks (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 unique nodes, i.e., \( |V| \times d \), where each row \( x_v \) 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 \( X^t \) across time steps \( t = 0, 1, \ldots, T \); and, (2) the change of graph structure \( G^t = (V, E^t) \) across time. For brevity, we use \( V' \) to denote all unique nodes, i.e., \( V' = \bigcup_{t=0}^{T} V^t \), so the change in \( G^{t} \) is reflected as the change of \( E^t \). 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 \( (G^0, X^0), (G^1, X^1), \ldots, (G^T, X^T) \) of both the graph structure and node attributes. Each single snapshot \( (G^t, X^t) \) 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 \( D = \{(G^t, X^t) | t = 0, 1, \ldots, T\} \), learn a mapping function \( f(D) : V \times \{0, 1, \ldots, T\} \to \mathbb{R}^d \) that embeds each node \( v \in V \) into a \( d \)-dimensional (typically \( d \ll |V| \)) representation vector \( h_v^t \) 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 $H^t$ should contain information not only about the current snapshot $(G^t, X^t)$, but also summarize the co-evolution trend from recent past into near future. Specifically, we aim at learning $H^t$ 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 $(G^{t-5}, X^{t-5}), \ldots, (G^{t-1}, X^{t-1})$.
- Being highly indicative about the developing co-evolution of node attributes and graph structure of next $S$ graph snapshots in future $(G^{t+1}, X^{t+1}), \ldots, (G^{t+S}, X^{t+S})$.

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 $\{(G^t, X^t) | t = 0, \ldots, T\}$, CoEvoGNN’s weight matrices $\{W(s) | s = 1, \ldots, S\}$ and its fusion matrix $\Gamma$, the temporal evolution span $S$, and a set of static models $\{f_{static}(s) | s = 1, \ldots, S\}$. CoEvoGNN first generates the initial latent embedding of node from the leading graph snapshot $(G^0, X^0)$ (Line 3 of Alg 1). In practice, we can use an arbitrary static GNN algorithm (e.g., GCN [11], GAT [24] and GRAPHASE [7]) as $f_{static}(s)$ functions.

We will examine the choice of $f_{static}(s)$ in Section 5. In particular, we concatenate the intermediate node embeddings at different structural depths together, i.e., $\hat{h}^{(s)}_v = f_{static}(v | G, X, L) \in \mathbb{R}^{dL_1}$, 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 $f_{static}$ 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 Alg 1). The newly fused $h^t_v$ gets l2 normalized and returned as the output node latent embedding (Line 20 in Alg 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 $f_{static}$ as its underlying aggregator, CoEvoGNN is able to distill structural information from each single time step independently. This means the resulting node embeddings $H^t$ are solely determined by its corresponding graph snapshot $(G^t, X^t)$, 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 $H^{t-1}$ into the current one $H^t$ [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 $H^t$ depend on all previous node embeddings $H^0, \ldots, H^{t-1}$, 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 $f_{static}$ to obtain its rich neighbor structural information $h^{(s)}_v$ (where $(\cdot)$ 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 Alg 1). Each one of these neighbor structural information embeddings $h^{(s)}_v \in \mathbb{R}^{dL_1}$ is also processed into the pre-attention

```
Algorithm 1: CoEvoGNN framework

Input : Dynamic graph sequence $\{(G^t, X^t) | t = 0, \ldots, T\}$; parameter matrices $\{W(s) | s = 1, \ldots, S\}$ and fusion matrix $\Gamma$; temporal evolution span $S$; and, static graph neural models $\{f_{static}(s) | s = 1, \ldots, S\}$.
Output : Node latent embeddings $h^t_v, v \in V$ and $1 \leq t \leq T$.
1 for $v \in V$ do
  2 // Initialization
  3 $h^{(1)}_v \leftarrow f_{static}(v | (G^0, X^0), 1)$
  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 $\hat{h}^{(s)}_v \leftarrow f_{static}(v | (G^{t-s}, X^{t-s}), L)$
  9 $\hat{e}^{(s)}_v \leftarrow (\hat{h}^{(s-1)}_v)^\top \cdot \Gamma \cdot \hat{h}^{(s)}_v$
  10 $\hat{H}_v[s] = \hat{h}^{(s)}_v$ and $E_v[s] = \hat{e}^{(s)}_v$
  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 $d^{(s)}_v \leftarrow \exp(\Sigma_{s'=1}^{\min(t, S)} \exp(E_v[s'])) / \Sigma_{s'=1}^{\min(t, S)} \exp(E_v[s'])$
  16 $A_v[s] = d^{(s)}_v$
  17 end
  18 // Fusion and normalization
  19 $h^t_v \leftarrow \Sigma_{s=1}^{\min(t, S)} A_v[s] \sigma(W(s) \cdot [h^{(s-1)}_v; \hat{H}_v[s]])$
  20 $h^t_v \leftarrow h^t_v / \|h^t_v\|_2$
  21 end
```
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 \( \hat{h}_t \) over training time steps \( t = 1, \ldots, T \). In a forward pass, for each mini-batch of nodes \( \mathcal{V}' \subseteq \mathcal{V} \), the result embeddings gets evaluated by the overall loss. During backpropagation, we use stochastic gradient descent to update the set of weight matrices \( \{W^{(s)} \mid s = 1, \ldots, S \} \), the fusion matrix \( \Gamma \), and attribute transformation matrix \( M \) (see Section 4.2.1), which parameterizes the proposed CoEvoGNN model.

\[
\min_{\hat{h}_t, v \in \mathcal{V}', t=1, \ldots, T} \mathcal{F} = \sum_{t=1}^{T} \sum_{v \in \mathcal{V}'} \alpha J_{v'}(\hat{h}_t) + (1-\alpha) J_{G'}(\hat{h}_t). \tag{1}
\]

This multi-task evolutionary objective is mainly composed of two terms: the attribute evolutionary loss \( J_{v'} \), and the structure evolutionary loss \( J_{G'} \). A mixture hyperparameter \( \alpha \) is used to balance the magnitude of these two terms.

4.2.1 Attribute evolutionary loss for attribute inference. The attribute evolutionary loss \( J_{v'} \) is defined as below:

\[
J_{v'}(\hat{h}_t) = \|\sigma(M \cdot \hat{h}_t) - x^{'(t)}_v\|^2_F, \tag{2}
\]

where \( M \) is the attribute transformation matrix and \( \sigma \) is non-linear function such as ReLU or sigmoid. Given a node latent embedding \( h^{'(t)}_t \in \mathbb{R}^{d \times 1} \), the attribute transformation matrix \( M \in \mathbb{R}^{r \times d} \) is used for mapping \( h^{'(t)}_t \) back into the \( r \)-dim raw attribute space. Node \( v \)'s remapped attribute inference vector \( \sigma(M \cdot \hat{h}_t) \in \mathbb{R}^{r \times 1} \) is then compared against the true node attribute vector \( x^{'(t)}_v \) by measuring the L2 distance. Note that parameter matrix \( M \), 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 \( J_{G'} \) is defined as below:

\[
J_{G'}(h^{'(t)}_t) = -\log \left( \sigma \left( (h^{'(t)}_t)^T \cdot \hat{h}_t \right) \right) - Q \cdot \mathbb{E}_{u' \sim P(u)} \log \left( \sigma \left(- (h^{'(t)}_t)^T \cdot h^{'(t)}_u \right) \right), \tag{3}
\]
Table 1: On co-authorship attributed graph sequence, CoEvoGNN outperforms baselines on forecasting $X^{2010}$ and $G^{2010}$.

| Method      | Attributes $X^{2010}$ | Links in $G^{2010}$ | Attributes $X^{2010}$ | Links in $G^{2010}$ |
|-------------|----------------------|---------------------|----------------------|---------------------|
|             | 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    | 0.939   | 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

1. Initialize model parameters $\{W^{(s)}|s = 1, \ldots, S\}$, $\Gamma$, and $M$.
2. Repeat
3. Sample minibatch of nodes $V'$ from all nodes $V$.
4. $H^1, \ldots, H^T \leftarrow \text{CoEvoGNN}(V')$ \quad \text{(see Algorithm 1)}
5. // Compute evolutionary losses
6. $J_1, \ldots, J_T \leftarrow \text{Compute the attribute evolutionary loss for attribute inference} \quad \text{(see Equation (2))}$
7. $J_1', \ldots, J_T' \leftarrow \text{Compute the structure evolutionary loss for link prediction} \quad \text{(see Equation (3))}$
8. $J \leftarrow \text{Compute overall loss} \quad \text{(see Equation (1))}$
9. // Update parameters
10. $W^{(s)} \leftarrow -\nabla_{W^{(s)}}(J)$
11. $\Gamma \leftarrow -\nabla_{\Gamma}(J)$
12. $M \leftarrow -\nabla_{M}(J)$
13. 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 $\sigma$ 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 $f_{\text{static}}^{(s)}(\cdot)$ is $O\left(\prod_{l=1}^{L} s_l \right)$ 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 $O\left( T \prod_{l=1}^{L} s_l \right)$. 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 $D_{\text{au}}^{2010}$ and $D_{\text{au}}^{10K}$. 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 $D_{\text{au}}^{2010}$; and 448 venues, 6,442 words in $D_{\text{au}}^{10K}$.

**Evolutionary virtual currency graph.** We used 2 benchmark datasets Bitcoin-OTC and Bitcoin-Alpa of Bitcoin transaction networks [12] denoted by $D_{\text{bc}}^{448}$ and $D_{\text{bc}}^{4K}$. 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 methods 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 output 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 $D_{AU}^{2K}$ and $D_{AU}^{10K}$. 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 $D_{bc}^{opt}$ and $D_{bc}^{alp}$.

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 us 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.

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 neighboring 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.
