CPDG: A Contrastive Pre-Training Method for Dynamic Graph Neural Networks

Yuanchen Bei[1*], Hao Xu[2*], Sheng Zhou[1,3†], Huixuan Chi[4],
Haishuai Wang[1], Mengdi Zhang[2], Zhao Li[1], Jiajun Bu[1]

[1] Zhejiang Provincial Key Laboratory of Service Robot, College of Computer Science and Technology, Zhejiang University
[2] Meituan   [3] School of Software Technology, Zhejiang University
[4] Institute of Computing Technology, Chinese Academy of Science
yuanchenbei@zju.edu.cn, kingsleyhsu1@gmail.com, zhousheng_zju@zju.edu.cn, chihuixuan21s@ict.ac.cn,
haishuai.wang@gmail.com, mdzhangmd@gmail.com, zhao_li@zju.edu.cn, bjj@zju.edu.cn

Abstract—**Dynamic graph data mining has gained popularity in recent years due to the rich information contained in dynamic graphs and their widespread use in the real world. Despite the advances in dynamic graph neural networks (DGNNs), the rich information and diverse downstream tasks have posed significant difficulties for the practical application of DGNNs in industrial scenarios. To this end, in this paper, we propose to address them by pre-training and present the Contrastive Pre-Training Method for Dynamic Graph Neural Networks (CPDG). CPDG tackles the challenges of pre-training for DGNNs, including *generalization* capability and *long-short term modeling* capability, through a flexible structural-temporal subgraph sampler along with structural-temporal contrastive pre-training schemes. Extensive experiments conducted on both large-scale research and industrial dynamic graph datasets show that CPDG outperforms existing methods in dynamic graph pre-training for various downstream tasks under three transfer settings.**

Index Terms—**dynamic graph neural networks, pre-training, contrastive learning**

I. INTRODUCTION

Graph is ubiquitous in real-world scenarios and mining graph data has attracted increasing attention from both academic and industry communities [1]–[3], such as epidemiology [4], [5], bioinformatic [6], [7], and recommender system [8]–[10]. Previous graph mining works have mainly focused on *static* graphs, neglecting the *dynamic* nature of real-world graph data, especially in industrial scenarios [11]. For example, in Meituan[1], users engage in real-time interactions with various items and shops, forming a rapidly changing large-scale dynamic graph. Such dynamic and evolving patterns are essential for accurate graph modeling [12], which has gained growing interest in practice.

Recently, the dynamic graph neural networks (DGNNs) have achieved success for dynamic graph mining [12]–[17]. A typical DGNNs workflow trains the model on dynamic graph data and then directly applies it for inference [11]. However, the learned patterns from training may not always be applicable to new inference graphs, especially in dynamic and large-scale industrial systems. Furthermore, frequent retraining is

especially impractical in large-scale industrial systems, where billions of interactions may occur within a short time interval. On the one hand, DGNNs require simultaneous consideration of temporal and structural information, resulting in higher computational complexity compared to traditional graph neural networks (GNNs) for static graphs. On the other hand, the unique requirements of different tasks in dynamic graphs, such as node-level or graph-level and temporal or structural, further limit the possibility of retraining for each new task. To sum up, *rich information and various downstream tasks have posed significant difficulties for the application of DGNNs*.

In recent years, the *pre-training and fine-tuning* strategy has been widely studied in natural language processing [18], computer vision [19] and gradually expanded to the graph domain [20]–[24]. The goal of graph pre-training is to learn the transferable knowledge representation on large-scale graph data, which can be applied to various downstream applications and tasks. Coincidentally, such a strategy has the potential to address the aforementioned challenges in applying DGNNs to real-world systems through pre-training on historical dynamic graph data, followed by fine-tuning for specific downstream tasks, rather than retraining from scratch.

Although feasible, the dynamic nature has brought extra challenges to pre-training DGNNs compared with static graph neural networks: First, **Generalization Capability**. Generalization is fundamental but essential for pre-training strategies [25], particularly in cases where dynamic information brings various downstream tasks that involve both temporal and structural aspects. However, current DGNNs are mostly trained for specific tasks like dynamic link prediction and lack the ability to generalize to various downstream tasks [12], [16]. Second, **Long-Short Term Modeling Capability**. In dynamic graphs, both long-term stable patterns and short-term fluctuating patterns are important natures and urgent for downstream tasks, especially for rapidly changing industrial dynamic graphs. Existing DGNNs have focused on modeling consistent changes over time to capture the long-term stable temporal patterns by utilizing memory buffers or temporal smoothing operators [16], [26]. However, when pre-training on graphs with large time intervals, the short-term fluctuating patterns can easily be overshadowed by long-term stable

---

* Both authors contributed equally to this paper.
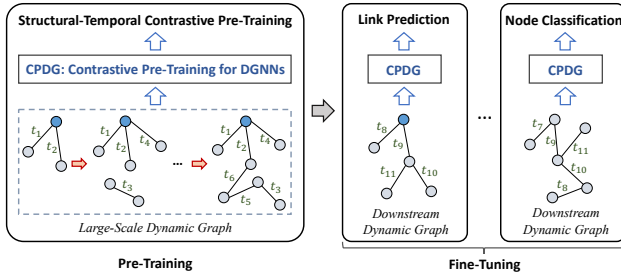*† Corresponding author.*
[1]https://meituan.com

Fig. 1. Overall pre-training and fine-tuning workflow of the proposed CPDG.

TABLE I
CHARACTERISTICS COMPARISON BETWEEN REPRESENTATIVE
PRE-TRAINING METHODS AND OUR PROPOSED CPDG.

Key Properties	Dynamic Data	Long-term Stable Pattern	Short-term Fluctuating Pattern	Pre-training Strategy
DGI [27]	✗	✗	✗	Contrastive
GPT-GNN [22]	✗	✗	✗	Generative
PT-DGNN [28]	✓	✗	✗	Generative
DDGCL [26]	✓	✓	✗	Contrastive
SelfRGNN [29]	✓	✓	✗	Contrastive
CPDG (ours)	✓	✓	✓	Contrastive

patterns. How to simultaneously capture both long-short term patterns during DGNN pre-training is still under-explored.

To tackle the above challenges, in this paper, we propose **CPDG**, a novel Contrastive Pre-Training Method for Dynamic Graph Neural Networks. Figure 1 illustrates the overall workflow of our proposed CPDG method. More specifically, CPDG first proposes a structural-temporal sampler that extracts informative subgraph with flexible temporal-aware sampling probability. Additionally, both the temporal and structural contrastive pre-training schemes are carefully designed to learn transferable long-short term evolution patterns and discriminative structural patterns in dynamic graphs. To further benefit the downstream tasks, CPDG explicitly extracts the evolved patterns during pre-training and provides for downstream fine-tuning. We conduct extensive experiments on both large-scale real-world dynamic graph datasets and the industrial dataset in Meituan. The results demonstrate that CPDG outperforms state-of-the-art graph pre-training methods on all datasets under various transfer settings and downstream tasks. Further experimental studies of CPDG prove the effectiveness of the designed method from multiple dimensions. The main contributions of this paper are summarized as follows.

- We highlight the difficulties of applying DGNNs in industrial scenarios, that the model complexity and various tasks make it impractical to retrain DGNNs. We propose to address them with pre-training for DGNNs which has received limited attention in the literature.
- We propose a novel CPDG method to tackle the challenges of pre-training and efficiently learn the transferable knowledge on large-scale dynamic graphs by a novel flexible structural-temporal subgraph sampler along with two views of subgraph contrastive pre-training schemes.
- We conduct extensive experiments on both large-scale public dynamic graph datasets and the industrial dataset in Meituan with different transfer settings and downstream tasks. Experimental results demonstrate the effectiveness and the generalization ability of CPDG.

In the following sections, we will first review the previous work related to our method in Section II. Second, we will give some key preliminaries of our work in Section III. Then, the detailed description of the proposed CPDG will be introduced in Section IV. To further verify the effectiveness of CPDG, we conduct various experiments in Section V. Finally, the conclusion of this paper is posed in Section VI.

II. RELATED WORKS

A. Dynamic Graph Neural Networks

Graph Neural Networks (GNNs) have shined a light on graph-structured data mining in recent years to guide the representation learning [2], [30]–[34]. However, since these networks are mainly designed for *static graphs*, they lack the ability to simultaneously capture the temporal and structural patterns within dynamic graphs, which are more common in the real world. Therefore, Dynamic Graph Neural Networks (DGNNs) have gained increasing attention recently, and many methods have been proposed and achieved great success in dynamic graph representation learning [12]–[17], [35].

Among them, DyRep [15] posits dynamic graph representation learning as a latent mediation process and utilizes the temporal point process for dynamic graph modeling. JODIE [14] designs a coupled recurrent model to learn dynamic node embeddings with the update, projection, and prediction components. TGAT [12] further designs a self-attention temporal-topological neighborhood aggregator for dynamic graph modeling. Then, TGN [16] designs a generic and state-of-the-art framework for dynamic graph learning with the memory module and unifies most of the above methods into this framework.

B. Pre-Training for GNNs

With the continuous success of GNNs in graph mining and the trend of increasing graph data scale in recent years, a growing number of works have begun to note the important role of graph pre-training and explore various solutions for pre-training GNNs on unannotated graph data [20]–[24], [36].

Most current works focus on static graphs. Among them, a direct way is to design some unsupervised tasks to let GNNs (e.g. GraphSAGE [32] and GAT [33]) pre-train on unlabeled data, such as link prediction [37]. Another category of methods is graph contrastive learning-based pre-training: DGI [27] maximizes the mutual information between node representations and graph summaries. Recently, GCC [21] proposes a contrastive learning scheme on subgraph perspectives with an instance discrimination task to learn transferable universal structural patterns. Besides, GPT-GNN [22] designs a generative graph pre-training paradigm with masked node attribute generation and edge generation tasks inspired by the pre-train language model.

In recent years, few works have begun to pay attention to pre-training DGNNs on dynamic graphs. PT-DGNN [28] improves GPT-GNN with temporal-aware masking. DDGCL [26] maximizes the time-dependent agreement between a node identity's two temporal views. Recently, SelfRGNN [29] proposes a Riemannian reweighting self-contrastive approach for self-supervised learning on dynamic graphs. We illustrate and compare the key characteristics between our designed CPDG and these state-of-the-art pre-training methods in Table I.

III. PRELIMINARIES

In this section, we present the key definitions related to pre-training for DGNNs. The main symbols are listed in Table II.

A. Dynamic Graph

There exist two main types of dynamic graphs, *discrete-time dynamic graphs* (DTDG) and *continuous-time dynamic graphs* (CTDG) [11]. DTDG is a sequence of static graph snapshots taken at intervals in time. CTDG is more general and can be represented as temporal lists of events, which reflects more detailed temporal signals and fine-grained rather than the coarse-grained DTDG. In this paper, we focus on the CTDG that is widely used in industrial systems and more demand for pre-training, which can be formulated as:

Definition 1. *Dynamic Graph [16] is defined as $\mathcal{G} = (\mathcal{V}^T, \mathcal{E}^T)$, where $\mathcal{V}^T$ is a temporal set of vertices, $\mathcal{E}^T$ is the temporal set of edges, and T is the time set. $N = |\mathcal{V}^T|$ denotes the number of vertices in $\mathcal{G}$. Each edge $e_{i,j}^t \in \mathcal{E}^T$ is denoted as a triple, $e_{i,j}^t = (i, j, t)$, where node $i, j \in \mathcal{V}^T$ and time $t \in T$. Each (i, j, t) means node i has an interaction with node j at time t. We denote the temporal graph $\mathcal{G}$ at time t as the graph $\mathcal{G}^t = (\mathcal{V}^t, \mathcal{E}^t)$, $\mathcal{V}^t$ and $\mathcal{E}^t$ as the set of vertices and edges observed before time t. We denote $\mathcal{N}_i^t = \{j \mid e_{i,j}^{t^-} \in \mathcal{E}^t, t^- \le t\}$ as the neighborhood set of node i in time interval $[0, t]$, and $\mathcal{N}_i^{k,t}$ as the set of k-hop neighborhoods of node i.*

B. Dynamic Graph Neural Networks

Given a dynamic graph $\mathcal{G} = (\mathcal{V}^T, \mathcal{E}^T)$, the existing Dynamic Graph Neural Network (DGNN) encoder learns the temporal embeddings for all nodes at time t, denoted as $\mathbf{Z}^t = (\mathbf{z}_1^t, ..., \mathbf{z}_i^t, ...)$. Besides, a memory $\mathcal{M}$ stores the memory state for each node at time t, denoted as $\mathbf{S}^t = (\mathbf{s}_1^t, ..., \mathbf{s}_i^t, ...)$, in which each state memorizes the temporal evolution of the node at the time interval $[0, t]$ in compressed format. The paradigm of DGNN encoder can be formulated as a function of node i and its k-hop temporal neighbors $\mathcal{N}_i^{k,t}$ at time t:

$$\mathbf{z}_i^t = \text{Emb}(i, t) = \sum_{u \in \mathcal{N}_i^{k,t}} f\left(\mathbf{s}_i^t, \mathbf{s}_u^{t^-}\right), \qquad (1)$$

where $f(\cdot)$ denotes a learnable function [16], such as dynamic graph attention. $\mathbf{s}_i^t$ denotes the state of node i that is stored in memory at time t, and $\mathbf{s}_u^{t^-}$ is the latest state of node u before time t, which is initialized as a zero vector for new encountered nodes and updated with batch processing by three

TABLE II
MAIN SYMBOLS AND DEFINITIONS IN THE PAPER.

Notations	Description
$\mathcal{G} = (\mathcal{V}^T, \mathcal{E}^T)$	A dynamic graph.
$\mathcal{V}^T$	The temporal set of vertices.
$\mathcal{E}^T$	The temporal set of edges.
T	The time set.
$N = |\mathcal{V}^T|$	The number of vertices.
$\mathcal{G}^t = (\mathcal{V}^t, \mathcal{E}^t)$	The dynamic graph at time t.
$\mathcal{V}^t$	The set of vertices observed before time t.
$\mathcal{E}^t$	The set of edges observed before time t.
$\mathcal{N}_i^t$	The neighborhood set of node i before time t.
$\mathcal{N}_i^{k,t}$	The k-hop neighborhoods of node i before time t.
$\mathbf{Z}^t$	The temporal embeddings at time t.
$\mathbf{S}^t$	The memory slots at time t.
$\mathbf{m}_i^t$	A message to node i at time t.
T_i^t	The set of event time that contains node i by time t.

TABLE III
IMPLEMENT FOR $f(\cdot)$, $\text{Msg}(\cdot)$, $\text{Agg}(\cdot)$ AND $\text{Mem}(\cdot)$ OF DIFFERENT DGNN ENCODERS.

Model	$f(\cdot)$	$\text{Msg}(\cdot)$	$\text{Agg}(\cdot)$	$\text{Mem}(\cdot)$
JODIE [14]	Time	Identity	–	RNN
DyRep [15]	Identity	Attention	–	RNN
TGN [16]	Attention	Identity	Last-time	GRU

following steps: *Message Function*, *Message Aggregator* and *Memory Updater*.

(i) Message Function. Given an interaction (i, j, t) involving node i, a message is computed to update state $\mathbf{s}_i^t$ of node i in the memory at time t, which can be expressed as:

$$\mathbf{m}_i^t = \text{Msg}\left(\mathbf{s}_i^{t^-}, \mathbf{s}_j^{t^-}, \phi(\Delta t)\right), \qquad (2)$$

where $\text{Msg}(\cdot)$ is the message function, such as identity and MLP [16]. Δt denotes the last updated time interval between $\mathbf{s}_i^{t^-}$ and $\mathbf{s}_j^{t^-}$, $\phi(\cdot)$ represents a generic time encoding [12].

(ii) Message Aggregator. As each node i may have multiple interaction events in time interval $[t_1, t]$ and each event generates a message, a message aggregator is further used to aggregate all the messages $\mathbf{m}_i^{t_1}, ..., \mathbf{m}_i^{t_b}$ for $t_1, ..., t_b \le t$:

$$\overline{\mathbf{m}}_i^t = \text{Agg}\left(\mathbf{m}_i^{t_1}, ..., \mathbf{m}_i^{t_b}\right), \qquad (3)$$

where $\text{Agg}(\cdot)$ is an aggregation function, such as mean and last time aggregation [16].

(iii) Memory Updater. The state $\mathbf{s}_i^t$ of node i at time t is updated upon $\overline{\mathbf{m}}_i^t$ and its previous state $\mathbf{s}_i^{t^-}$, which can be formulated as:

$$\mathbf{s}_i^t = \text{Mem}\left(\mathbf{s}_i^{t^-}, \overline{\mathbf{m}}_i^t\right), \qquad (4)$$

where $\text{Mem}(\cdot)$ represents a time series function, such as RNN [38], LSTM [39] and GRU [40].

Note that most popular DGNN encoders have followed the above paradigm, such as JODIE [14], DyRep [15] and TGN [16]. They differ in the implementation of $f(\cdot)$, $\text{Msg}(\cdot)$, $\text{Agg}(\cdot)$ and $\text{Mem}(\cdot)$, as compared in Table III following the organization in previous works [16].
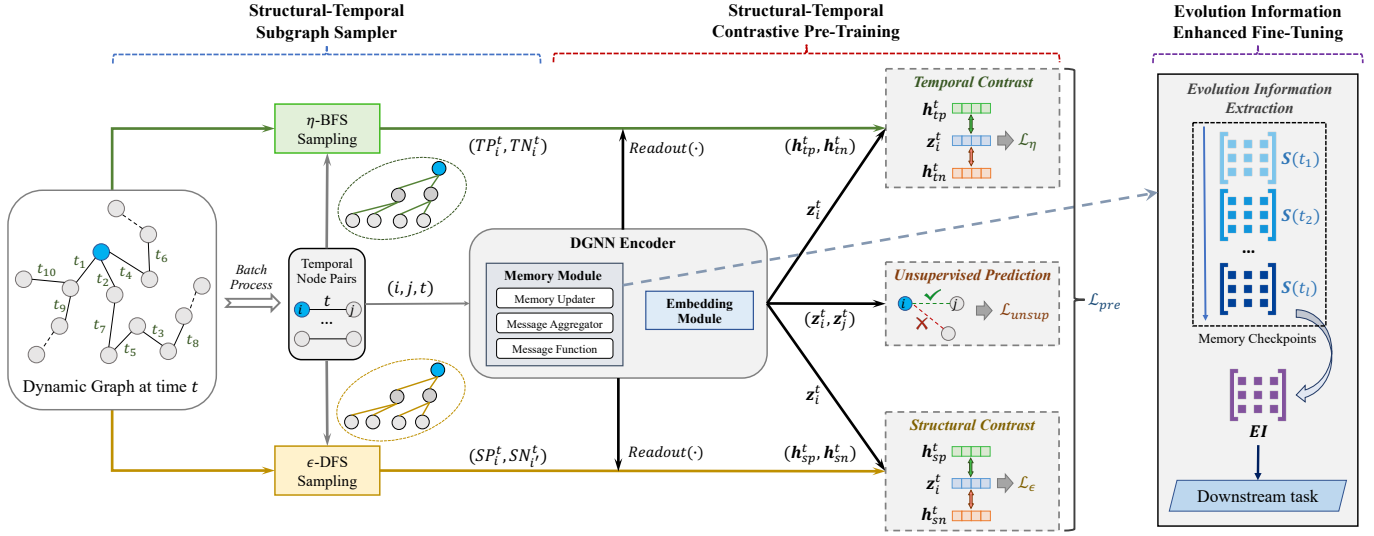
Fig. 2. The overall framework of the proposed CPDG model. CPDG contains three major parts: (i) the structural-temporal subgraph sampler which obtains the positive-negative subgraph pairs with the temporal preference by η-BFS sampling and with the structural preference by ϵ-DFS sampling; (ii) structural-temporal contrastive pre-training which mines the informative and transferable long-short term temporal patterns from temporal contrast and structural patterns from structural contrast; (iii) evolution information enhanced fine-tuning which is an optional module that can assist fine-tuning in downstream tasks with the pre-trained evolution information.

Definition 2. *Pre-Training for Dynamic Graph Neural Networks* *aims at pre-training a generalizable DGNN encoder* f_θ *on unlabeled large-scale dynamic graph* $\mathcal{G} = (\mathcal{V}^T, \mathcal{E}^T)$ *in a self-supervised way so that* f_θ *can benefit initialization of models in downstream tasks. It is worth noting that in the dynamic graph downstream tasks, the newly encountered events may contain nodes that occur in the pre-training stage, thus it is also beneficial to extract the temporal pattern of these nodes and propagate them in the fine-tuning stage.*

IV. METHODOLOGY

In this section, we introduce our proposed **CPDG**, which consists of the following parts: Firstly, a flexible *structural-temporal subgraph sampler* is applied with various sampling strategies to obtain subgraphs with different preferences. More specifically, we provide a η-BFS sampling strategy and a ϵ-DFS sampling strategy for extracting the subgraph with temporal preference and structural preference. Secondly, with the proposed sampler, a *structural-temporal contrastive pre-training* module is further proposed with both temporal contrast and structural contrast to learn transferable long-short term evolution patterns and discriminative structural patterns simultaneously. Finally, an optional *evolution information enhanced fine-tuning* module is further proposed to extract informative long-short term evolution patterns during pre-training and provide for downstream tasks. Figure 2 illustrates the overall network architecture of CPDG.

A. Structural-Temporal Subgraph Sampler

To obtain informative and transferable information for model pre-training, we design two sampling strategies for different focuses rather than uniformly random sampling

schemes in most existing methods [12], [16]. The structural-temporal subgraph sampler is designed to flexibly access various structural/temporal-aware subgraph sampling functions on the dynamic graph data for extracting subgraphs with structural-temporal preferences and providing contrastive pre-training with informative samples.

η-*BFS Sampling Strategy*: Existing works on DGNNs have successfully modeled the long-term stable patterns within dynamic graphs by utilizing memory buffers or temporal smoothing restrictions [16], [26], [29]. However, in addition to the long-term stable patterns that can be modeled by current DGNN models, the short-term fluctuating patterns are also of great significance for real-life industrial graphs with rapid changes. In real-world recommender systems, both the long-term and short-term interests of users extracted from the user-item interaction graph are important for making optimal recommendation decisions [41], [42]. For instance, in the Meituan platform, besides the enduring stable trend of users' personal taste preferences, popular factors contribute to the rapid daily fluctuations in user interests [43]. In order to further capture the short-term fluctuating temporal evolution patterns in dynamic graphs during pre-training, we design a η-BFS sampling strategy to extract temporal subgraphs by accessing various temporal-aware sampling functions.

We first define the set of event time that contains node i by time t as $T_i^t = \{t_u | (i, u, t_u) \in \mathcal{E}^t, t_u < t\}$, given a root node i at time t, the η-BFS sampling first observes all the 1-hop neighbors $u \in \mathcal{N}_i^t$ of node i. A sampling probability p_u is then assigned to each neighbor u by a temporal-aware function $f_{t \to p}(\cdot)$ under T_i^t. The 1-hop η-neighbors are randomly sampled from $\mathcal{N}_i^t$ depending on p_u. By conducting the above sampling process on each sampled 1-hop η-neighbors, we get the 2-hop η-neighbors. After recursively conducting the above
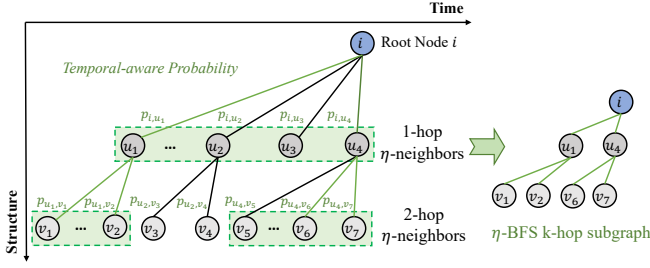
Fig. 3. A toy example of η-BFS sampling with $\eta = 2$ and $k = 2$. Node u_1 and u_4 are first selected as 1-hop η-neighbors, $\{v_1, v_2, v_6, v_7\}$ are then sampled from the neighborhoods of u_1 and u_4 as 2-hop η-neighbors, the output of η-BFS sampling is a subgraph consisting of $\{i, u_1, u_4, v_1, v_2, v_6, v_7\}$.

Fig. 4. A toy example of ϵ-DFS sampling with $\epsilon = 2$ and $k = 2$. Node u_4 and u_5 are the latest interacted two nodes and thus selected as 1-hop ϵ-neighbors, $\{v_5, v_6, v_7, v_8\}$ are the latest interacted nodes of $\{u_4, u_5\}$ and selected as 2-hop ϵ-neighbors, the output of ϵ-DFS sampling is a subgraph consisting of $\{i, u_4, u_5, v_5, v_6, v_7, v_8\}$.

sampling for k times, the η-BFS sampling generates the η-BFS k-hop subgraph. Figure 3 illustrates a toy example of η-BFS sampling with $\eta = 2$ and $k = 2$. The η-BFS sampling strategy will be utilized with two designed temporal-aware probability functions to generate sample pairs for the following temporal contrastive learning in Subsection IV-B.

ϵ-DFS Sampling Strategy: In addition to the temporal evolution pattern, the unique and discriminative structural pattern of each node is also indispensable for the pre-training of DGNNs [21]. A naive way to capture the unique structural pattern is conducting a random walk starting from node i to generate the subgraph [44], which is equivalent to the depth-first search (DFS) rooted by node i. However, such vanilla DFS will miss the temporal order that evolved over time in dynamic graphs. We further extend the vanilla DFS with temporal-aware selection and propose a structural ϵ-DFS sampler for maintaining both structural and temporal characteristics.

Following the definition of $T_i^t = \{t_u | (i, u, t_u) \in \mathcal{E}^t, t_u < t\}$ in Subsection IV-A, we chronologically sort all the 1-hop neighbors from $\mathcal{N}_i^t$ as $\mathcal{NS}_i^t$, and select the most recent interacted ϵ neighbors:

$$\mathcal{NS}_i^t = [a, b, ..., \underbrace{u, ..., v, ...}_{\epsilon-\text{neighbors}}], \qquad (5)$$

where $t_a \leq t_b \leq ... \leq t_u \leq t_v \leq ... < t$. It is worth noting that such expansion only considers the most recent temporal information, which is consistent with the η-BFS but in a discrete formulation. Then, for each selected 1-hop neighbor, we add it to subgraph $\mathcal{G}_i^t$ and repeat the above sampling process for k times. Figure 4 illustrates a toy example of ϵ-DFS sampler with $\epsilon = 2$ and $k = 2$. The ϵ-DFS sampling strategy will be utilized to generate sample pairs for the following structural contrastive learning in Subsection IV-B.

Since both η-BFS and ϵ-DFS samplers are independent of the learnable parameters or node embeddings during the pre-training stage, the sampling operators can be preprocessed before pre-training for efficiency.

B. Structural-temporal Contrastive Pre-training

With the help of the proposed structural-temporal subgraph sampler with the designed η-BFS sampling strategy and ϵ-DFS sampling strategy, the structure-temporal contrastive pre-training is designed to capture the informative and transferable
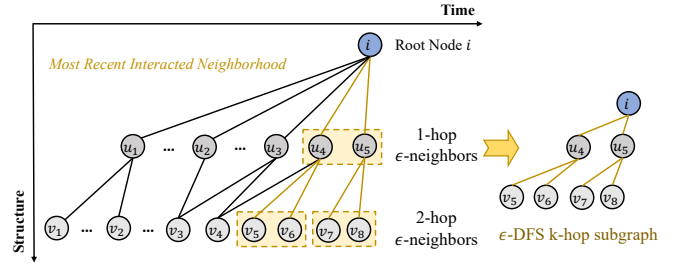
long-short term evolution patterns and structural patterns from these samples.

Temporal Contrast (TC): The DGNN backbone encoders with the memory module have the ability to learn long-term stable patterns [14]–[16]. Nevertheless, the short-term fluctuating evolution patterns over time are also vital for DGNNs' real-world applications, such as user modeling in above mentioned industrial scenarios. Thus, short-term fluctuations should also be highlighted during the modeling at each timestamp. Therefore, the temporal contrast is proposed by contrasting the recent subgraph (positive sample) with the agelong subgraph (negative sample) to capture the short-term fluctuating patterns over time changing. The motivation for temporal contrast is that the relatively recent events are more consistent with the current state than the agelong events.

Concerning the consistency of temporal information, the temporal-aware function $f_{t \to p}(\cdot)$ can be implemented in two opposite ways for generating the positive and negative samples: *chronological probability function $f_{t \to p}^{tp}(\cdot)$* and *reverse chronological probability function $f_{t \to p}^{tn}(\cdot)$*. The probabilities are defined as follows:

Chronological Probability is proportional to the time interval between the event time t_u and t:

$$\hat{t}_u = \frac{t_u - \min T_i^t}{t - \min T_i^t}, \qquad (6)$$

$$p_u^{tp} = \frac{\exp\left(\hat{t}_u / \tau\right)}{\sum_{v \in \mathcal{N}_i^t} \exp\left(\hat{t}_v / \tau\right)}, \qquad (7)$$

where τ is the temperature coefficient.

Reverse Chronological Probability is inversely proportional to the time interval between the event time t_u and t:

$$p_u^{tn} = \frac{\exp\left(\widetilde{t}_u / \tau\right)}{\sum_{v \in \mathcal{N}_i^t} \exp\left(\widetilde{t}_v / \tau\right)}, \qquad (8)$$

where $\widetilde{t}_u = 1 - \hat{t}_u$ and $\hat{t}_u$ is defined in Eq. (6).

Given an interaction event (i, j, t), CPDG first obtains the embedding z_i^t of node i at time t with DGNN encoder. Then, CPDG generates temporal positive and negative subgraphs by η-BFS sampling strategy with the chronological and reverse chronological probability, denoted as $\mathcal{TP}_i^t$ and $\mathcal{TN}_i^t$. For each node u in $\mathcal{TP}_i^t$ and node v in $\mathcal{TN}_i^t$, we retrieve the

embeddings from the memory $\mathcal{M}$ and then pool them into a subgraph-wise embedding with a readout function:

$$\mathbf{h}_{tp}^t = \text{Readout}\left(\mathbf{s}_u^t, u \in \mathcal{TP}_i^t\right), \quad (9)$$

$$\mathbf{h}_{tn}^t = \text{Readout}\left(\mathbf{s}_v^t, v \in \mathcal{TN}_i^t\right), \quad (10)$$

where $\text{Readout}(\cdot)$ is a kind of graph pooling operation [45], such as min, max, and weighted pooling. In this paper, we use mean pooling for simplicity.

The temporal contrastive learning is implemented with a triplet margin loss [46] among center node embedding z_i^t, positive temporal subgraph embedding $\mathbf{h}_{tp}^t$ and negative temporal subgraph embedding $\mathbf{h}_{tn}^t$:

$$\mathcal{L}_\eta = \frac{1}{|\mathcal{V}^t|} \sum_{i \in \mathcal{V}^t} \mathbb{E}\left\{\max\{d\left(\mathbf{z}_i^t, \mathbf{h}_{tp}^t\right) - d\left(\mathbf{z}_i^t, \mathbf{h}_{tn}^t\right) + \alpha, 0\}\right\}, \quad (11)$$

where α is the margin constant, $d(\cdot)$ is the distance metric and we adopt Euclidean distance in this paper.

Note that the long-term stable patterns are captured by the memory module of the DGNN encoder. The temporal contrast and states from the memory module in DGNN collaboratively capture the long-short term evolution patterns.

Structural Contrast (SC): Besides the informative temporal evolution patterns, discriminative structural patterns are also important for structure modeling on dynamic graphs.

Following the instance discrimination paradigm, we treat the subgraph generated by ϵ-DFS sampler of node i as the positive sample, and the subgraph generated by ϵ-DFS sampler of another random node $i' \neq i$ as the negative sample for structural contrastive learning.

Analogously, after obtaining the embedding z_i^t of node i at time t, CPDG generates structural positive and negative subgraphs by S-ϵ-DFS sampler, denoted as $\mathcal{SP}_i^t$ and $\mathcal{SN}_{i'}^t$. For each node u in $\mathcal{SP}_i^t$ and node v in $\mathcal{SN}_i^t$, we retrieve the embeddings from the memory $\mathcal{M}$ and then pool them into a subgraph-wise embedding with a readout function:

$$\mathbf{h}_{sp}^t = \text{Readout}\left(\mathbf{s}_u^t, u \in \mathcal{SP}_i^t\right), \quad (12)$$

$$\mathbf{h}_{sn}^t = \text{Readout}\left(\mathbf{s}_v^t, v \in \mathcal{SN}_{i'}^t\right). \quad (13)$$

The structural contrastive learning is implemented with a triplet margin loss among center node embedding z_i^t, positive structural subgraph embedding $\mathbf{h}_{sp}^t$ and negative structural subgraph embedding $\mathbf{h}_{sn}^t$:

$$\mathcal{L}_\epsilon = \frac{1}{|\mathcal{V}^t|} \sum_{i \in \mathcal{V}^t} \mathbb{E}\left\{\max\{d\left(\mathbf{z}_i^t, \mathbf{h}_{sp}^t\right) - d\left(\mathbf{z}_i^t, \mathbf{h}_{sn}^t\right) + \alpha, 0\}\right\}, \quad (14)$$

where $d(\cdot)$ is the same as Eq.(11) in temporal contrast.

Overall Pre-Training Objective Function: Following the general setting of unsupervised dynamic graph representation learning [12], [16], CPDG finally adds an auxiliary temporal link prediction pre-text task for pre-training. Given a pair of temporal edge (i, j, t), the affinity score is calculated by:

$$\hat{y}_{i,j}^t = \sigma\left(\text{MLP}(\mathbf{z}_i^t \| \mathbf{z}_j^t)\right), \quad (15)$$

where $\sigma(\cdot)$ denotes the sigmoid function. The temporal link prediction task is optimized by a binary cross-entropy loss [47] as follows:

$$\mathcal{L}_{tlp} = \sum_{(i,j,j',t) \in \mathcal{O}} -y_{i,j}^t \cdot \log(\hat{y}_{i,j}^t) + y_{i,j'}^t \cdot \log(\hat{y}_{i,j'}^t), \quad (16)$$

where $\mathcal{O} = \{(i, j, j', t)|(i, j, t) \in \mathcal{E}^t, (i, j', t) \notin \mathcal{E}^t\}$, and $y_{i,j}^t \in \{0, 1\}$ donates whether there is an edge between node i and j at time t.

Overall, the objective function of CPDG method can be formulated as a linear combination of $\mathcal{L}_\eta$, $\mathcal{L}_\epsilon$ and $\mathcal{L}_{tlp}$:

$$\mathcal{L}_{pre} = (1 - \beta) \cdot \mathcal{L}_\eta + \beta \cdot \mathcal{L}_\epsilon + \mathcal{L}_{tlp}. \quad (17)$$

where $\beta \in (0, 1)$ is a hyperparameter to balance the impact of temporal contrast and structure contrast. The CPDG method with any DGNN encoders is pre-trained using the gradient descent on the objective function $\mathcal{L}_{pre}$.

C. Evolution Information Enhanced Fine-Tuning

In the CPDG pre-training, the memory $\mathcal{M}$ stores the long-short term evolution information of each node through the designed pre-training objections, which is beneficial for the downstream dynamic graph tasks. For example, if node i occurs in both the pre-training and fine-tuning stages, the evolution pattern of node i can be propagated to the neighborhoods in the downstream task for further enhancement.

With the above findings, we further design this evolution information enhanced fine-tuning (short as EIE) module as an optional auxiliary scheme for downstream fine-tuning. During the pre-training of CPDG, we uniformly store L checkpoints of the memory $\mathcal{M}$ and fuse the sequence of checkpoints as the evolution information:

$$\mathbf{EI} = f_{EI}\left([\mathbf{S}^1, ..., \mathbf{S}^l, ..., \mathbf{S}^L]\right), \quad (18)$$

where $\mathbf{S}^l$ denotes l-th memory checkpoint from pre-training, $f_{EI}(\cdot)$ can be any kind of sequence operation function, such as mean pooling, attention mechanism, and GRU [40].

In the fine-tuning stage, we first transform the evolution information $\mathbf{EI}$ with a two-layer MLP to let it better fit to the downstream data, then we combine it with the downstream temporal embeddings $\mathbf{Z}^{\text{down}}$ as the enhanced embeddings:

$$\mathbf{Z}^{EIE} = [\mathbf{Z}^{\text{down}} \| \text{MLP}(\mathbf{EI})], \quad (19)$$

where $[\cdot \| \cdot]$ is the matrix concatenation operator. The fine-tuning in downstream tasks is then conducted on the enhanced embeddings $\mathbf{Z}^{EIE}$.

D. Model Analysis

Pre-training stage: We define the time complexity of DGNN as $O(D)$, where $O(D)$ varies from different DGNN backbones. To pre-train the model scalably, the Monte Carlo trick [48] is widely utilized for DGNNs for model training under batch processing, which is also suitable for CPDG. For CPDG, the additional time consumption depends on sampling and contrastive pre-training. For sampling temporal contrastive subgraph pair of N nodes, the time complexity is $O(2\eta^k N)$,

where η and k denote sampling width and depth respectively. For sampling structural contrastive subgraph pair of N nodes, the time complexity is $O(2\epsilon^k N)$, where ϵ and k also denote sampling width and depth. We set $\eta = 10$, $\epsilon = 10$, and $k = 2$ as the default setup so that $\eta^k \ll N$ and $\epsilon^k \ll N$. Then, the contrastive learning equipped with a non-parameter mean-pooling operation has the time complexity of $O(4N)$. Therefore, the overall time complexity of CPDG pre-training is $O(D+2(\eta^k+\epsilon^k+2)N)$. The pseudocode of the pre-training procedure of CPDG is described as the given Algorithm 1.

Fine-tuning stage: If we fine-tune CPDG without the EIE strategy (i.e. full fine-tune), the complexity is the same as only the DGNN backbone with $O(D)$. Then, if we fine-tune CPDG with EIE enhancement, the complexity comes from Eq.(18) and Eq.(19), of which the complexity of Eq.(18) depends on the selected operation function of $f_{EI}(\cdot)$, and the complexity of Eq.(19) is $O(N)$. Therefore, the complexity of optional fine-tuning strategies can be summarized as Table IV, where L is the pre-trained checkpoint length and we set it as 10 for default. The performance comparison results with different fine-tuning ways are also conducted in the following Subsection V-G. The procedure of evolution information enhanced fine-tuning in CPDG is illustrated at Algorithm 2.

Algorithm 1 The Pre-Training Procedure of CPDG

Input: Dynamic graph $\mathcal{G} = (\mathcal{V}^T, \mathcal{E}^T)$; η-BFS sampler $f_\eta(\cdot)$; ϵ-DFS sampler $f_\epsilon(\cdot)$; Hyperparameters β; Pre-training epochs P.

Output: Pre-trained DGNN model parameters θ^* for downstream fine-tuning.

1: Randomly initialize the DGNN model f_θ and the trainable parameters in CPDG.
2: **for** $epoch \in 1, 2, ...P$ **do**
3:     **for all** $(i, j, t) \in \mathcal{G}$ sorted by timestamp **do**
4:         Obtain temporal contrast sample pair $\mathcal{TP}_i^t$ and $\mathcal{TN}_i^t$ by $f_\eta(\cdot)$;
5:         Obtain structural contrast sample pair $\mathcal{SP}_i^t$ and $\mathcal{SN}_i^t$ by $f_\epsilon(\cdot)$;
6:         Conduct the temporal contrast learning with the sampled $\mathcal{TP}_i^t$ and $\mathcal{TN}_i^t$, and calculate the temporal contrast loss $\mathcal{L}_\eta$ via Eq.(11);
7:         Conduct the structural contrast learning with the sampled $\mathcal{SP}_i^t$ and $\mathcal{SN}_i^t$, and calculate the temporal contrast loss $\mathcal{L}_\epsilon$ via Eq.(14);
8:         Conduct the auxiliary pre-text task and calculate the corresponding loss $\mathcal{L}_{tlp}$ via Eq.(16);
9:     **end for**
10:     Minimize the joint pre-training loss $\mathcal{L}_{pre} = (1 - \beta) \cdot \mathcal{L}_\eta + \beta \cdot \mathcal{L}_\epsilon + \mathcal{L}_{tlp}$;
11:     Update model's learnable parameters by using stochastic gradient descent;
12:     *// Optional for fine-tuning*
13:     Save the checkpoint from the memory $\mathcal{M}$ in f_θ;
14: **end for**
15: Save the pre-trained DGNN parameters θ^*.

Fine-tuning Strategy	Complexity
CPDG-(w/o EIE)	$O(D)$
CPDG-(EIE-mean)	$O(D + N + 1)$
CPDG-(EIE-attn)	$O(D + 2N)$
CPDG-(EIE-GRU)	$O(D + N + NL^2)$

Algorithm 2 The Evolution Information Enhanced Fine-Tuning Procedure of CPDG

Input: Pre-trained DGNN model f_{θ^*}; Dynamic graph $\mathcal{G} = (\mathcal{V}^T, \mathcal{E}^T)$; Sequence of evolution checkpoints $[\mathbf{S}^1, ..., \mathbf{S}^L]$; Evolution information fusion function $f_{EI}(\cdot)$; Fine-tuning epochs F.

Output: Model performance on the downstream task.

1: **for** $epoch \in 1, 2, ...F$ **do**
2:     Obtain downstream node embeddings $\mathbf{Z}^{down}$ by the pre-trained DGNN model f_{θ^*};
3:     Fuse $[\mathbf{S}^1, ..., \mathbf{S}^L]$ by $f_{EI}(\cdot)$ and obtain the evolution information $\mathbf{EI}$;
4:     Obtain the enhanced embeddings $\mathbf{Z}^{EIE}$ with $\mathbf{Z}^{down}$ and $\mathbf{EI}$ via Eq.(19);
5:     Conduct the downstream task with $\mathbf{Z}^{EIE}$ and obtain the corresponding fine-tuning loss $\mathcal{L}_{down}$;
6:     Minimize $\mathcal{L}_{down}$ and fine-tune DGNN's parameters.
7: **end for**

V. EXPERIMENTS

In this section, we conduct extensive experiments on five public widely-used dynamic graph datasets and an industrial dynamic graph dataset from Meituan to demonstrate the effectiveness of the proposed CPDG method from various perspectives. To evaluate the generalization ability of CPDG, we consider three *transfer settings* and two *downstream tasks*. 1) Transfer settings include time transfer, field transfer, and time+field transfer. 2) Downstream tasks include both *dynamic link prediction* and *dynamic node classification*.

A. Experimental Datasets

The experiments are conducted on five widely-used research datasets and an industrial dataset from Meituan on various downstream tasks. The details of these datasets are introduced as follows.

Dynamic Link Prediction Datasets:

Amazon Review[2] [49] is a large-scale user-product dynamic graph dataset containing 20.9 million users, 9.3 million products, and 82.8 million reviews (edges) into 29 product fields. The time span of these reviews is in the range of May 1996 - Oct 2018. We select the *Beauty*, *Luxury*, and *Arts, Crafts, and Sewing* fields for the experiment. The *user-product interaction prediction* is set as the downstream dynamic link prediction task for Amazon dataset.

---

[2]https://jmcauley.ucsd.edu/data/amazon

Gowalla[3] [50] is a famous social network for users check-in at various locations, containing about 36 million check-ins made by 0.32 million users over 2.8 million locations. These check-in records are in the time span of Jan 2009 - June 2011. The locations are grouped into 7 main fields. We select the *Entertainment*, *Outdoors*, and *Food* fields for the experiment. The *check-in prediction* is set as the downstream dynamic link prediction task for Gowalla dataset.

The data split of Amazon and Gowalla varies from different transfer settings considering both time and field, which is detailed described in Subsection V-C.

Meituan is a *industrial* user-poi interaction (i.e. click and purchase) dataset collected from Meituan food delivery service in Beijing of China from Feb.14th to Mar.28th, 2022 (42 days), which contains about 0.75 million interaction records (sorted by time). Each interaction record contains the user-ID, poi-ID, timestamp, and other contexts. We utilize this industrial dataset for evaluating the dynamic link prediction task with a ratio of 6:4 sorted by time for pre-training and downstream.

Dynamic Node Classification Datasets:

Wikipedia[4] [14] is a dynamic network between users and edited pages for node classification. It contains 9,227 nodes and around 157,474 temporal edges. Dynamic labels indicate banned users.

MOOC[5] [14] is a dynamic network of students and online courses for node classification, which contains 7,145 nodes and 411,749 interactions between them. Dynamic labels indicate drop-out students.

Reddit[6] [14] is a dynamic graph between active users and their posts under subreddits for node classification, with about 11,000 nodes and 672,447 temporal edges, and dynamic labels indicating banned users.

We split each dynamic node classification dataset with the ratio of 6:2:1:1 sorted by time for pre-training, downstream training, validation, and testing, respectively.

B. Baselines

We compare the proposed CPDG method with ten representative methods, including state-of-the-art static graph learning methods and dynamic graph learning methods, which can be organized into four main categories:

Task-supervised Static Graph Learning Models:

- **GraphSAGE** [32] learns node representation by sampling and aggregating features from the node's local neighborhood without temporal information. It performs link prediction as its pre-training task.
- **GAT** [33] aggregates the neighborhood message with the multi-head attention mechanism. The pre-training task of GAT is the same as GraphSAGE.
- **GIN** [51] is a simple GNN architecture that generalizes the Weisfeiler-Lehman test. It also performs the link prediction task for the model pre-training.

Self-supervised Static Graph Learning Models:

- **DGI** [27] maximizes the mutual information between local patch representation and corresponding global graph summary in a self-supervised manner.
- **GPT-GNN** [22] is a state-of-the-art generative pre-training framework with self-supervised node attribute generation and edge generation tasks on large-scale graph pre-training.

Task-supervised Dynamic Graph Learning Models:

- **DyRep** [15] is a temporal point process (TPP-based) dynamic graph model, which posits representation learning as a latent mediation process. Following its task setting for dynamic graphs, we adopt temporal link prediction as its pre-training task.
- **JODIE** [14] is a state-of-the-art dynamic graph model, which employs two recurrent neural networks and a novel projection operator to estimate the embedding of a node at any time in the future. The pre-training task of JODIE is the same as DyRep.
- **TGN** [16] is a generic and state-of-the-art dynamic graph learning framework with memory modules and dynamic graph-based operators. The pre-training task of TGN is also the same as DyRep.

Self-supervised Dynamic Graph Learning Models:

- **DDGCL** [26] is a self-supervised dynamic graph learning method via contrasting two nearby temporal views of the same node identity, with a time-dependent similarity critic and GAN-type contrastive loss.
- **SelfRGNN** [29] is a state-of-the-art self-supervised Riemannian dynamic graph neural network with the Riemannian reweighting self-contrast for dynamic graph learning.

For all the baselines, we first pre-train them on the same pre-training data as CPDG and utilize them for initialization with full fine-tuning strategy [21] in downstream tasks.

C. Experimental Setup

To evaluate the effectiveness of the CPDG method, we follow the experimental setting of popular existing pre-training work [22] with three *transfer settings* and two *downstream tasks* in fine-tuning on these datasets.

Transfer Settings:

- **Time transfer**. We split the graph by time spans for pre-training and fine-tuning on the same field. For the Amazon Review dataset, we split the data before *2017* for pre-training and data since *2017* for fine-tuning. For the Gowalla dataset, we split the data before *2011* for pre-training and data since *2011* for fine-tuning. For Meituan, Wikipedia, MOOC, and Reddit datasets, due to the lack of field categories, we only conduct the *time transfer* and adopt the same chronological data split with the first 60% for pre-training and the rest for fine-tuning.
- **Field transfer**. We pre-train the model from one graph field and transfer it to other graph fields. For the Amazon Review dataset, we use data from *Arts, Crafts, and Sewing* field for pre-training and data from *Beauty* and *Luxury* fields for fine-tuning. For the Gowalla dataset, we use data from the

TABLE V
DETAILED STATISTICS OF THE EXPERIMENTAL DATASETS WITH BOTH TIME AND FIELD INFORMATION. (T: TIME TRANSFER, F: FIELD TRANSFER, T+F: TIME+FIELD TRANSFER).

Dataset		Amazon					Gowalla				
Data Description		Data Field	Data Range	# Nodes	# Edges	Density	Data Field	Data Range	# Nodes	# Edges	Density
Pre-train	T	Beauty	May 1996-Dec 2016	261,889	270,714	0.0008%	Entertainment	Jan 2009-Dec 2010	208,309	223,777	0.0010%
		Luxury	May 1996-Dec 2016	303,542	401,987	0.0009%	Outdoors	Jan 2009-Dec 2010	223,777	1,192,387	0.0047%
	F	Arts, Crafts, and Sewing	Jan 2017-Oct 2018	729,282	897,079	0.0003%	Food	Jan 2011-June 2011	673,858	2,708,688	0.0011%
	T+F	Arts, Crafts, and Sewing	May 1996-Dec 2016	1,345,428	1,978,838	0.0002%	Food	Jan 2009-Dec 2010	679,586	5,067,244	0.0021%
Downstream	T/F/T+F	Beauty	Jan 2017-Oct 2018	107,916	100,631	0.0017%	Entertainment	Jan 2011-June 2011	199,547	733,863	0.0036%
		Luxury	Jan 2017-Oct 2018	143,686	172,641	0.0013%	Outdoors	Jan 2011-June 2011	227,980	763,764	0.0029%

TABLE VI
DETAILED STATISTICS OF THE DATASETS WITHOUT FIELD INFORMATION.

Dataset	# Nodes	# Edges	# Timespan	Density
Meituan	~45,000	~750,000	42 days	0.0741%
Wikipedia	9,227	157,474	30 days	0.3700%
MOOC	7,145	411,749	30 days	1.6133%
Reddit	11,000	672,447	30 days	1.1116%

Food field for pre-training and data from *Entertainment* and *Outdoors* fields for fine-tuning.

- **Time+Field transfer**. In this setting, we consider graph transfer from different time spans and fields simultaneously. That is, we use the data from fields before a particular time for model pre-training and the data from other fields after a particular time for fine-tuning. The split time and fields are the same as the above two tasks: For the Amazon dataset, we first pre-train the model on *Arts, Crafts, and Sewing* field under the data range from May 1996 to Dec 2016, and then we fine-tune the DGNN model on *Beauty* or *Luxury* data fields under the data range from Jan 2017 to Oct 2018. For the Gowalla dataset, we pre-train the model on *Food* field under the data range from Jan 2009 to Dec 2010, then we fine-tune the DGNN model on *Entertainment* or *Outdoors* fields under the data range from Jan 2011 to June 2011.

Downstream Task Settings:

- **Dynamic link prediction**. For Amazon Review, Gowalla, and Meituan datasets, we aim to predict future user-item interactions at a specific time as the downstream task. Then, we adopt the widely used AUC and AP metrics as the evaluation metrics. Note that we conduct the downstream task in *Beauty* and *Luxury* fields for the Amazon Review dataset, and in *Entertainment* and *Outdoors* fields for Gowalla.
- **Dynamic node classification**. For Wikipedia, MOOC, and Reddit datasets, we aim to predict the node state with a specific class label at a specific time as the downstream task. Then, we adopt the widely used AUC metric as the node classification evaluation metric.

We introduce the statistics and settings of the experimental datasets in Table V and Table VI.

Environment Configuration and Hyper-parameter Setting:

The model hyper-parameters are optimized via grid search on all the experimental datasets, and the best models are selected by early stopping based on the AUC score on the validation set. Hyper-parameter ranges of the model for grid search are the following: learning rate in $\{5 \times 10^{-4}, 1 \times 10^{-4}, 5 \times 10^{-3}, 1 \times 10^{-3}, 5 \times 10^{-2}, 1 \times 10^{-2}\}$, the batch size settings

are searched from range $\{256, 512, 1024, 2048, 4096, 8192\}$ for the link prediction task due to large-scale property, and the batch size of the node classification task is set to 512. All weighting matrices are initialized by Xavier initialization for all models. For all dynamic graphs (Dyrep, JODIE, TGN, DDGCL, SelfRGNN, CPDG), the memory states are all initialized with zero. Note that we run all the experiments *five* times with different random seeds and report the average results with standard deviation to prevent extreme cases. All experiments are conducted on NVIDIA A100 (80G) GPUs.

D. Main Results

In this section, we report the comparison results between CPDG and the baselines for various downstream tasks, including dynamic link prediction and node classification. The hyper-parameters of CPDG are set as $\eta = 10$, $\epsilon = 10$, $k = 2$, and $L = 10$ in the main results, and we will analyze the impact of changes in these hyper-parameters in the Subsection V-H.

Performance on Dynamic Link Prediction Task:

The results and comparisons on Amazon Review and Gowalla datasets under three transfer settings are shown in Table VII. And the results and comparisons on Meituan under the time transfer setting are shown in Table VIII. From the results, we have the following observations.

- **Our proposed CPDG achieves the best performance under different transfer settings.** Specifically, the average improvements over the best-performed baseline on the AP metric of all the fields on Amazon Review and Gowalla datasets are 1.59%, 1.33%, 1.60% w.r.t. *time transfer*, *field transfer*, and *time+field transfer* settings, respectively. Furthermore, in the Meituan industrial dataset, compared to the DGNN models, the performance gains by CPDG can also be observed. These observations demonstrate the effectiveness of our CPDG through flexible structural-temporal samplers along with temporal-view and structural-view subgraph contrastive pre-training schemes and optional evolution information enhanced fine-tuning scheme.
- **The dynamic graph methods generally perform significantly better than the static graph methods.** Among the compared baselines, the dynamic graph methods (DyRep, JODIE, TGN, DDGCL, and SelfRGNN) generally perform significantly better than the static graph methods (Graph-SAGE, GIN, GAT, DGI, and GPT-GNN) on all the datasets. This further verifies the necessity of capturing the temporal evolution patterns in the dynamic graphs. Another interesting finding is that in dynamic graph tasks, the static

TABLE VII

COMPARISON RESULTS ON DYNAMIC LINK PREDICTION TASK. THE BEST AND SECOND-BEST RESULTS IN EACH COLUMN ARE HIGHLIGHTED IN **BOLD** FONT AND <u>UNDERLINED</u>. NAN DENOTES NOT A NUMBER OF THE VALUE OF LOSS FUNCTION DURING THE PRE-TRAINING STAGE.

	Downstream Dataset	Amazon				Gowalla			
	Evaluation Task	Beauty		Luxury		Entertainment		Outdoors	
	Evaluation Metric	AUC	AP	AUC	AP	AUC	AP	AUC	AP
Time Transfer	GraphSAGE	0.7537±0.0062	0.6896±0.0053	0.6395±0.0040	0.5814±0.0028	0.6315±0.0018	0.6906±0.0009	0.6183±0.0007	0.6797±0.0003
	GIN	0.6908±0.0100	0.6904±0.0073	0.5948±0.0014	0.5815±0.0014	0.5179±0.0015	0.6189±0.0007	0.5154±0.0021	0.6213±0.0016
	GAT	0.5217±0.0015	0.6123±0.0023	0.5403±0.0012	0.6292±0.0020	0.5315±0.0005	0.6242±0.0007	0.5420±0.0032	0.6334±0.0024
	DGI	0.6928±0.0047	0.6773±0.0018	0.6083±0.0017	0.5805±0.0024	0.5763±0.0037	0.6541±0.0025	0.5955±0.0039	0.6675±0.0012
	GPT-GNN	0.5785±0.0010	0.5650±0.0018	0.5532±0.0008	0.5267±0.0008	0.5139±0.0019	0.5157±0.0017	0.5118±0.0022	0.5090±0.0018
	DyRep	0.8023±0.0073	0.8152±0.0020	0.7853±0.0023	0.7906±0.0029	0.8490±0.0124	0.8586±0.0093	0.8269±0.0044	0.8337±0.0057
	JODIE	0.8472±0.0149	0.8411±0.0512	<u>0.8201±0.0030</u>	0.8147±0.0021	0.8572±0.0035	0.8669±0.0021	0.8274±0.0034	0.8390±0.0040
	TGN	0.8589±0.0025	0.8533±0.0016	<u>0.7985±0.0049</u>	0.7843±0.0049	<u>0.9152±0.0032</u>	<u>0.9076±0.0058</u>	<u>0.9051±0.0046</u>	<u>0.9073±0.0035</u>
	DDGCL	0.8146±0.0025	0.8474±0.0023	0.8066±0.0007	<u>0.8153±0.0015</u>	0.7117±0.0012	0.7586±0.0012	0.6617±0.0017	0.7087±0.0014
	SelfRGNN	0.6352±0.0253	0.6011±0.0312	0.5744±0.0405	0.5576±0.0371	0.5457±0.0113	0.5316±0.0107	0.5467±0.0192	0.5332±0.0164
	CPDG (DyRep)	0.8275±0.0028	0.8371±0.0020	0.7976±0.0031	0.8038±0.0037	0.8883±0.0040	0.8946±0.0043	0.8459±0.0020	0.8575±0.0001
	CPDG (JODIE)	<u>0.8672±0.0017</u>	**0.8665±0.0017**	**0.8378±0.0034**	**0.8390±0.0041**	0.8914±0.0010	0.8966±0.0004	0.8709±0.0004	0.8767±0.0008
	CPDG (TGN)	**0.8690±0.0026**	<u>0.8594±0.0022</u>	0.8042±0.0068	0.7931±0.0048	**0.9234±0.0019**	**0.9202±0.0042**	**0.9134±0.0026**	**0.9121±0.0045**
Field Transfer	GraphSAGE	0.7265±0.0232	0.6755±0.0145	0.6166±0.0067	0.5807±0.0046	0.6330±0.0012	0.6875±0.0006	0.6284±0.0025	0.6829±0.0012
	GIN	0.6652±0.0092	0.6516±0.0036	0.5782±0.0053	0.5651±0.0072	0.5167±0.0005	0.5886±0.0008	0.5176±0.0043	0.5598±0.0021
	GAT	0.5161±0.0120	0.5585±0.0243	0.5635±0.0380	0.5540±0.0195	0.5332±0.0004	0.6112±0.0005	0.5312±0.0006	0.6035±0.0011
	DGI	0.6922±0.0051	0.6408±0.0030	0.6027±0.0028	0.5731±0.0024	0.5724±0.0014	0.6479±0.0024	0.5849±0.0002	0.6505±0.0002
	GPT-GNN	0.5777±0.0013	0.5664±0.0020	0.5528±0.0006	0.5272±0.0009	0.5136±0.0017	0.5098±0.0011	0.5106±0.0016	0.5040±0.0009
	DyRep	0.8054±0.0023	0.7946±0.0073	0.7788±0.0047	0.7894±0.0098	0.8589±0.0077	0.8658±0.0057	0.8395±0.0057	0.8467±0.0063
	JODIE	0.8121±0.0027	0.8074±0.0042	0.7812±0.0039	0.7878±0.0048	0.8495±0.0090	0.8587±0.0075	0.8409±0.0027	0.8483±0.0030
	TGN	<u>0.8391±0.0052</u>	<u>0.8347±0.0033</u>	0.7753±0.0039	0.7628±0.0039	**0.8877±0.0050**	0.8788±0.0061	<u>0.8787±0.0033</u>	<u>0.8725±0.0054</u>
	DDGCL	0.7929±0.0021	0.8300±0.0023	<u>0.7854±0.0012</u>	<u>0.8014±0.0011</u>	0.7202±0.0009	0.7672±0.0007	0.6721±0.0008	0.7196±0.0009
	SelfRGNN	0.5313±0.0040	0.5315±0.0067	0.5140±0.0041	0.5110±0.0002	0.5051±0.0021	0.5097±0.0038	0.5123±0.0062	0.5106±0.0045
	CPDG (DyRep)	0.8124±0.0012	0.8226±0.0018	0.7827±0.0016	0.8008±0.0022	0.8659±0.0049	0.8742±0.0036	0.8512±0.0014	0.8543±0.0023
	CPDG (JODIE)	0.8220±0.0148	0.8255±0.0121	**0.8296±0.0032**	**0.8299±0.0017**	0.8734±0.0012	0.8781±0.0014	0.8516±0.0015	0.8573±0.0019
	CPDG (TGN)	**0.8439±0.0040**	**0.8372±0.0038**	0.7782±0.0030	0.7641±0.0065	<u>0.8870±0.0063</u>	**0.8800±0.0052**	**0.8868±0.0048**	**0.8841±0.0041**
Time+Field Transfer	GraphSAGE	0.7428±0.0047	0.6730±0.0034	0.6296±0.0047	0.5776±0.0044	0.5118±0.0141	0.5102±0.0090	0.5051±0.0109	0.5059±0.0120
	GIN	0.6696±0.0150	0.6543±0.0062	0.5854±0.0070	0.5741±0.0119	0.5089±0.0009	0.6053±0.0010	0.5111±0.0012	0.6089±0.0006
	GAT	0.5206±0.0115	0.5796±0.0080	0.5268±0.0027	0.5422±0.0153	0.5291±0.0005	0.6141±0.0014	0.5403±0.0007	0.6168±0.0012
	DGI	0.6846±0.0021	0.6524±0.0034	0.5990±0.0038	0.5757±0.0037	0.5714±0.0018	0.6374±0.0026	0.5843±0.0046	0.6525±0.0039
	GPT-GNN	0.5773±0.0016	0.5665±0.0006	0.5531±0.0005	0.5271±0.0008	0.5105±0.0020	0.5074±0.0022	0.5098±0.0030	0.5029±0.0029
	DyRep	0.8026±0.0087	0.8001±0.0067	0.7726±0.0064	0.7795±0.0074	0.8458±0.0113	0.8548±0.0092	0.8250±0.0038	0.8325±0.0041
	JODIE	0.8401±0.0199	0.8346±0.0165	<u>0.8115±0.0114</u>	<u>0.8124±0.0094</u>	0.8412±0.0080	0.8538±0.0069	0.8272±0.0063	0.8372±0.0059
	TGN	0.8478±0.0079	0.8449±0.0065	0.7820±0.0069	0.7739±0.0071	0.8622±0.0040	0.8607±0.0049	<u>0.8596±0.0078</u>	<u>0.8527±0.0105</u>
	DDGCL	0.8060±0.0009	0.8408±0.0009	0.8037±0.0009	0.8123±0.0019	0.7194±0.0009	0.7668±0.0006	0.6697±0.0015	0.7176±0.0011
	SelfRGNN	0.5374±0.0100	0.5586±0.0184	0.5156±0.0037	0.5144±0.0018	NaN	NaN	NaN	NaN
	CPDG (DyRep)	0.8113±0.0047	0.8115±0.0073	0.7746±0.0031	0.7901±0.0027	<u>0.8679±0.0073</u>	**0.8779±0.0056**	0.8350±0.0106	0.8487±0.0090
	CPDG (JODIE)	**0.8622±0.0031**	**0.8541±0.0033**	**0.8250±0.0028**	**0.8250±0.0034**	0.8667±0.0051	<u>0.8738±0.0027</u>	0.8405±0.0110	0.8490±0.0083
	CPDG (TGN)	<u>0.8597±0.0031</u>	<u>0.8523±0.0040</u>	0.7896±0.0026	0.7766±0.0035	**0.8732±0.0065**	0.8661±0.0090	**0.8720±0.0053**	**0.8678±0.0044**

generative graph pre-training framework performs relatively worse than other static GNNs, such as GraphSAGE, which has also been observed and discussed in [52].

- **The dynamic task-supervised methods generally perform better than dynamic self-supervised methods.** Generally, among the compared baselines, the dynamic task-supervised methods (DyRep, JODIE, and TGN) perform better than the dynamic self-supervised methods (DDGCL and SelfRGNN) on experimental datasets. This demonstrates the importance of memory in the DGNN encoder to capture the long-term evolution of each node in dynamic graphs and the insufficiency for general pattern modeling of current dynamic self-supervised learning manners under the pre-training setting.

Performance on Dynamic Node Classification Task:

In addition to the dynamic link prediction downstream task, we also conduct the dynamic node classification task

TABLE VIII

COMPARISON RESULTS ON THE MEITUAN INDUSTRY DATASET.

Dataset	Meituan	
Metric	AUC	AP
DyRep	0.8461±0.0028	0.8355±0.0023
CPDG (DyRep)	**0.8472±0.0061**	**0.8372±0.0033**
JODIE	0.8498±0.0042	0.8315±0.0077
CPDG (JODIE)	**0.8513±0.0042**	**0.8398±0.0034**
TGN	0.8431±0.0040	0.8304±0.0037
CPDG (TGN)	**0.8480±0.0037**	**0.8364±0.0047**

on Wikipedia, MOOC, and Reddit datasets compared with the state-of-the-art dynamic graph methods to further verify the generalization ability in various downstream tasks for CPDG. Table IX shows the node classification results. From these results, we have the following observation:

TABLE IX
COMPARISON RESULTS OF AUC METRICS ON DYNAMIC NODE
CLASSIFICATION TASK UNDER *time+transfer* SETTING.

Method	Wikipedia	MOOC	Reddit
DyRep	0.8189±0.0098	0.6342±0.0108	0.5614±0.0281
JODIE	0.8206±0.0067	0.6185±0.0049	0.5385±0.0303
TGN	0.8302±0.0012	**0.7009**±0.0029	0.5552±0.0106
DDGCL	0.7091±0.0162	0.5674±0.0061	0.5205±0.0207
SelfRGNN	0.8490±0.0024	0.6051±0.0027	0.5363±0.0051
CPDG (DyRep)	0.8323±0.0059	0.6689±0.0014	<u>0.6110</u>±0.0101
CPDG (JODIE)	**0.8649**±0.0023	0.6551±0.0019	0.5474±0.0089
CPDG (TGN)	<u>0.8558</u>±0.0024	<u>0.6797</u>±0.0254	**0.6551**±0.0118

- **CPDG also has the best performance under the dynamic node classification task in most cases**, which shows CPDG has good generalization capability in various downstream tasks. Note that the performance of CPDG is lower than TGN on the MOOC dataset. One reason is that the generic structure and temporal patterns are not as obvious in the MOOC dataset as in other datasets.
- **The dynamic task-supervised methods make a better performance than the dynamic self-supervised methods**. These results are consistent with the observation in dynamic link prediction, which further verifies the effectiveness of the memory module within DGNNs in learning long-term evolution. And the superiority of CPDG with generally all the equipped backbone DGNNs also illustrates the effectiveness of the model design for capturing other generic patterns.

E. Model Generalization

Backbone Generalization. To investigate the generalization of the proposed CPDG method, we evaluate the performance of CPDG with different DGNN encoders. We try three representative DGNN encoders namely, DyRep, JODIE, and TGN. More specifically, we experiment on each encoder being pre-trained with our CPDG and perform on all the datasets of both dynamic link prediction and dynamic node classification tasks. The results shown in Table VII, Table VIII, and Table IX *all illustrate the performance of pre-training different DGNN encoders with CPDG is consistently better than the vanilla task-supervised pre-training strategy*. This demonstrates both the effectiveness and generalization of the proposed CPDG method in different DGNN backbones.

Inductive Generalization. In order to further investigate the generalization of CPDG on inductive downstream tasks, we further compare the performance of CPDG with the JODIE encoder under the inductive link prediction task. The results in Table X demonstrate the performance of CPDG on three transfers under the inductive setting. We observe that CPDG significantly improves AUC under time transfer by about 10% on the Entertainment and Outdoors datasets, respectively. Moreover, under the time+field transfer, the most challenging transfer, CPDG still achieves 5.17% and 3.19% AUC gains on Entertainment and Outdoors datasets, respectively. Altogether, the results demonstrate that CPDG is also able to well generalize on the inductive downstream task.

TABLE X
INDUCTIVE STUDY RESULTS ON AUC AND AP METRICS.

Evaluation Metric		AUC	AP
Beauty	No Pre-train	0.6798±0.0106	0.6848±0.0220
	CPDG (T)	**0.7219**±0.0023 **(+6.19%)**	**0.7409**±0.0013 **(+8.19%)**
	CPDG (F)	0.6983±0.0081 (+2.72%)	0.7088±0.0099 (+3.50%)
	CPDG (T+F)	0.7026±0.0034 (+3.35%)	0.7201±0.0052 (+5.15%)
Luxury	No Pre-train	0.6927±0.0038	0.6991±0.0050
	CPDG (T)	**0.7187**±0.0023 **(+3.75%)**	**0.7358**±0.0046 **(+5.25%)**
	CPDG (F)	0.7100±0.0030 (+2.50%)	0.7267±0.0017 (+3.95%)
	CPDG (T+F)	0.7059±0.0034 (+1.91%)	0.7241±0.0055 (+3.58%)
Entertain	No Pre-train	0.7237±0.0130	0.7407±0.0115
	CPDG (T)	**0.8015**±0.0049 **(+10.75%)**	**0.8071**±0.0043 **(+8.96%)**
	CPDG (F)	0.7737±0.0036 (+6.91%)	0.7792±0.0043 (+5.20%)
	CPDG (T+F)	0.7611±0.0091 (+5.17%)	0.7714±0.0066 (+4.14%)
Outdoors	No Pre-train	0.7079±0.0036	0.7294±0.0034
	CPDG (T)	**0.7822**±0.0016 **(+10.50%)**	**0.7980**±0.0010 **(+9.40%)**
	CPDG (F)	0.7579±0.0025 (+7.06%)	0.7712±0.0029 (+5.73%)
	CPDG (T+F)	0.7356±0.0150 (+3.91%)	0.7551±0.0117 (+3.52%)

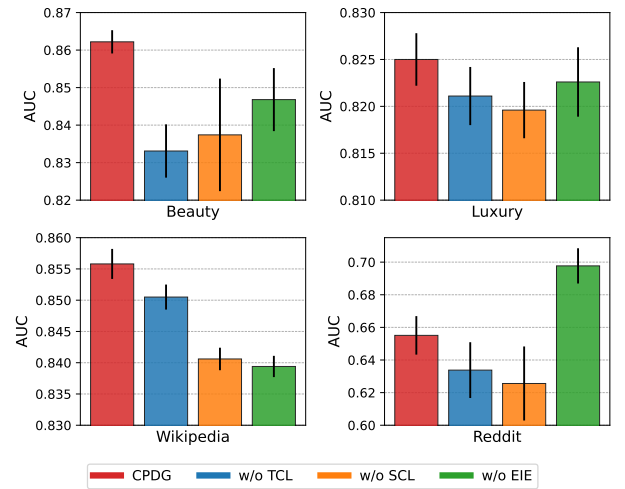

Fig. 5. Ablation study results between CPDG and its three variants on Amazon-Beauty, Amazon-Luxury, Wikipedia, and Reddit.

F. Ablation Study

The proposed CPDG method consists of three key carefully designed modules, including temporal contrast (TC), structural contrast (SC), and evolution information enhanced (EIE) fine-tuning. To shed light on the effectiveness of these modules, we conduct ablation studies on the variants of CPDG on both dynamic link prediction datasets (Beauty and Luxury) and dynamic node classification datasets (Wikipedia and Reddit). As shown in Figure 5, we report the results of CPDG without temporal contrast (w/o TC), without structural contrast (w/o SC), and without the evolution information enhanced fine-tuning (w/o EIE) of the Amazon Review dataset under the *time+field transfer* setting.

In general, we have the following findings: (i) *The performance of CPDG is significantly superior to the CPDG variants without TC, SC, or EIE in general* (w/o EIE achieve the better result on Reddit), which indicates the effectiveness of all three modules of the proposed CPDG method. (ii) Different datasets have different preferences for temporal and

TABLE XI
FINE-TUNING STRATEGIES COMPARISON RESULTS ON AMAZON-BEAUTY
AND AMAZON-LUXURY, UNDER *time+field transfer* SETTING.

Evaluation Metric		AUC	AP
Beauty	Full	0.8468±0.0084	0.8423±0.0077
	EIE-mean	0.8496±0.0048	0.8440±0.0049
	EIE-attn	0.8517±0.0031	0.8472±0.0026
	EIE-GRU	**0.8622**±0.0031	**0.8541**±0.0033
Luxury	Full	0.8226±0.0074	0.8213±0.0064
	EIE-mean	0.8237±0.0039	0.8244±0.0020
	EIE-attn	0.8201±0.0113	0.8214±0.0080
	EIE-GRU	**0.8250**±0.0028	**0.8250**±0.0034

structural patterns. The performance of CPDG w/o TC drops more than w/o SC on the Beauty field, indicating that temporal contrast provides more meaningful information for pre-training in this case. While the observation is just the opposite on the Luxury, Wikipedia, and Reddit fields. It suggests that different dynamic graphs have different emphases on temporal evolution information and structural evolution information, so it is meaningful for the CPDG to adopt both temporal and structural contrast collaboratively.

G. Discussion on EIE

In this subsection, we investigate the impact of the optional evolution information enhanced (EIE) fine-tuning strategy. There are different types of $f_{EI}(\cdot)$ that can be derived from different EIE variants, such as EIE-mean, EIE-attn, and EIE-GRU, which implement $f_{EI}(\cdot)$ with mean pooling, attention mechanism [53], and GRU operation, respectively.

The results on two fields of the Amazon dataset under the *time+field transfer* task with JODIE backbone are shown in Table XI, from which we can observe that: (i) **All the EIE variants generally perform better than the full fine-tuning strategy for downstream tasks**, even the simple mean pooling also has a good improvement compared to the full fine-tuning. This indicates that the information stored in pre-trained memory contains rich evolution information, which demonstrates the effectiveness of EIE for downstream tasks. (ii) **EIE-GRU performs best against other variants**. This observation demonstrates that the GRU operator has a better ability to capture the evolution pattern of pre-trained memory checkpoints. Due to the relatively high computational cost of GRU, the efficient EIE-mean can also be a suitable choice.

H. Parameter Study

Effect of the balanced parameter β. We investigate the impact of hyper-parameter β in Eq.(17) and report the results in Figure 6. We can observe that the performance of Beauty drops with the β increases in general, while that of Luxury is relatively stable. One possible reason is that temporal information is more essential in Beauty, while temporal information is almost as important as structural information in Luxury.

Effect of contrastive subgraph parameters η/ϵ **and** k. We investigate the combination effect of contrastive subgraph parameters η/ϵ and k as illustrated in Figure 7. We can observe
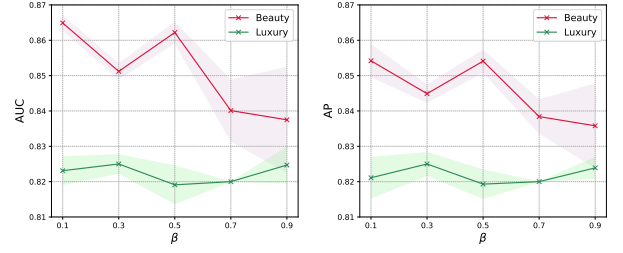

Fig. 6. Hyper-parameter study of different β value on Amazon-Beauty and Amazon-Luxury, under the *time+field transfer* setting.
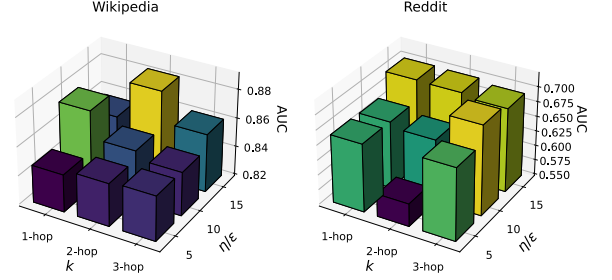

Fig. 7. Hyper-parameter study of different combination of k and η/ϵ.
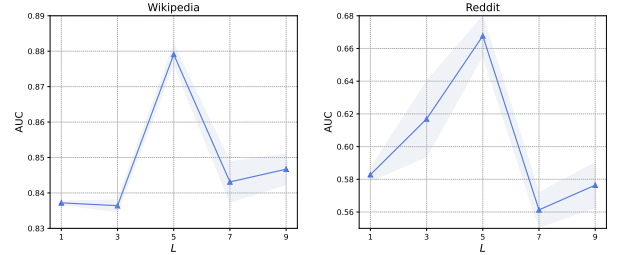

Fig. 8. Hyper-parameter study of different checkpoint length L in EIE.

that increasing η/ϵ (the width of the subgraph) generally has an improving impact on model performance, while deepening k (the depth of the subgraph) may not necessarily bring a positive effect on the performance.

Effect of checkpoint length L **in EIE**. We also explored the impact of L by varying it from 1 to 9 with the step size of 2 as shown in Figure 8. The results reveal that a checkpoint length of 5 yields relatively good performance while utilizing a length that is either too small or too large does not provide optimal benefits.

VI. CONCLUSION

In this paper, we propose to address the difficulties of applying DGNNs in industrial scenarios. We propose a novel CPDG method to capture the long-short term temporal evolution patterns and discriminative structural patterns through flexible structural-temporal subgraph samplers along with structural-temporal contrastive pre-training schemes. Furthermore, we introduce an optional evolution information enhanced fine-tuning strategy to take advantage of the evolved patterns during pre-training. Extensive experiments on widely used dynamic graph datasets and an industrial dataset in Meituan demonstrate the effectiveness of our proposed CPDG method. In future works, we will explore the CPDG in more practical scenarios, such as industrial recommender systems.

REFERENCES

[1] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proceedings of the 24th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2018, pp. 974–983.

[2] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4–24, 2020.

[3] Y. Bei, S. Zhou, Q. Tan, H. Xu, H. Chen, Z. Li, and J. Bu, "Reinforcement neighborhood selection for unsupervised graph anomaly detection," in *2023 IEEE International Conference on Data Mining (ICDM)*, 2023.

[4] R. Pastor-Satorras and A. Vespignani, "Epidemic spreading in scale-free networks," *Physical review letters*, vol. 86, no. 14, p. 3200, 2001.

[5] R. M. May and A. L. Lloyd, "Infection dynamics on scale-free networks," *Physical Review E*, vol. 64, no. 6, p. 066112, 2001.

[6] Y. Li, C. Huang, L. Ding, Z. Li, Y. Pan, and X. Gao, "Deep learning in bioinformatics: Introduction, application, and perspective in the big data era," *Methods*, vol. 166, pp. 4–21, 2019.

[7] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko *et al.*, "Highly accurate protein structure prediction with alphafold," *Nature*, vol. 596, no. 7873, pp. 583–589, 2021.

[8] S. Wu, F. Sun, W. Zhang, X. Xie, and B. Cui, "Graph neural networks in recommender systems: a survey," *ACM Computing Surveys*, vol. 55, no. 5, pp. 1–37, 2022.

[9] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "Lightgcn: Simplifying and powering graph convolution network for recommendation," in *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, 2020, pp. 639–648.

[10] Y. Bei, H. Chen, S. Chen, X. Huang, S. Zhou, and F. Huang, "Non-recursive cluster-scale graph interacted model for click-through rate prediction," in *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, 2023, pp. 3748–3752.

[11] J. Skarding, B. Gabrys, and K. Musial, "Foundations and modeling of dynamic networks using dynamic graph neural networks: A survey," *IEEE Access*, vol. 9, pp. 79 143–79 168, 2021.

[12] D. Xu, C. Ruan, E. Körpeoglu, S. Kumar, and K. Achan, "Proceedings of the inductive representation learning on temporal graphs," in *International Conference on Learning Representations*, 2020.

[13] G. H. Nguyen, J. B. Lee, R. A. Rossi, N. K. Ahmed, E. Koh, and S. Kim, "Continuous-time dynamic network embeddings," in *Companion proceedings of the web conference 2018*, 2018, pp. 969–976.

[14] S. Kumar, X. Zhang, and J. Leskovec, "Predicting dynamic embedding trajectory in temporal interaction networks," in *Proceedings of the 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2019, pp. 1269–1278.

[15] R. Trivedi, M. Farajtabar, P. Biswal, and H. Zha, "Dyrep: Learning representations over dynamic graphs," in *International Conference on Learning Representations*, 2019.

[16] E. Rossi, B. Chamberlain, F. Frasca, D. Eynard, F. Monti, and M. Bronstein, "Temporal graph networks for deep learning on dynamic graphs," in *ICML 2020 Workshop on Graph Representation Learning*, 2020.

[17] J. You, T. Du, and J. Leskovec, "Roland: graph learning framework for dynamic graphs," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 2358–2366.

[18] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019, pp. 4171–4186.

[19] S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, and M. Shah, "Transformers in vision: A survey," *ACM computing surveys (CSUR)*, vol. 54, no. 10s, pp. 1–41, 2022.

[20] W. Hu, B. Liu, J. Gomes, M. Zitnik, P. Liang, V. S. Pande, and J. Leskovec, "Strategies for pre-training graph neural networks," in *International Conference on Learning Representations*, 2020.

[21] J. Qiu, Q. Chen, Y. Dong, J. Zhang, H. Yang, M. Ding, K. Wang, and J. Tang, "Gcc: Graph contrastive coding for graph neural network pre-training," in *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2020, pp. 1150–1160.

[22] Z. Hu, Y. Dong, K. Wang, K.-W. Chang, and Y. Sun, "Gpt-gnn: Generative pre-training of graph neural networks," in *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2020, pp. 1857–1867.

[23] Y. Lu, X. Jiang, Y. Fang, and C. Shi, "Learning to pre-train graph neural networks," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, no. 5, 2021, pp. 4276–4284.

[24] M. Sun, K. Zhou, X. He, Y. Wang, and X. Wang, "Gppt: Graph pre-training and prompt tuning to generalize graph neural networks," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 1717–1727.

[25] Z. Liu, X. Yu, Y. Fang, and X. Zhang, "Graphprompt: Unifying pre-training and downstream tasks for graph neural networks," in *Proceedings of the ACM Web Conference 2023*, 2023, pp. 417–428.

[26] S. Tian, R. Wu, L. Shi, L. Zhu, and T. Xiong, "Self-supervised representation learning on dynamic graphs," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 1814–1823.

[27] P. Velickovic, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep graph infomax," in *International Conference on Learning Representations*, 2019.

[28] K.-J. Chen, J. Zhang, L. Jiang, Y. Wang, and Y. Dai, "Pre-training on dynamic graph neural networks," *Neurocomputing*, vol. 500, pp. 679–687, 2022.

[29] L. Sun, J. Ye, H. Peng, and P. S. Yu, "A self-supervised riemannian gnn with time varying curvature for temporal graph learning," in *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 2022, pp. 1827–1836.

[30] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[31] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2017.

[32] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[33] P. Velickovic, G. Cucurull, A. R. Arantxa Casanova, P. Lio, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations*, 2018.

[34] Y. Zhang, Y. Bei, S. Yang, H. Chen, Z. Li, L. Chen, and F. Huang, "Alleviating behavior data imbalance for multi-behavior graph collaborative filtering," in *2023 IEEE International Conference on Data Mining Workshops (ICDMW)*, 2023.

[35] Y. Wang, Y.-Y. Chang, Y. Liu, J. Leskovec, and P. Li, "Inductive representation learning in temporal networks via causal anonymous walks," in *International Conference on Learning Representations*, 2020.

[36] X. Jiang, T. Jia, Y. Fang, C. Shi, Z. Lin, and H. Wang, "Pre-training on large-scale heterogeneous graph," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2021, pp. 756–766.

[37] R. N. Lichtenwalter, J. T. Lussier, and N. V. Chawla, "New perspectives and methods in link prediction," in *Proceedings of the 16th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2010, pp. 243–252.

[38] L. R. Medsker and L. Jain, "Recurrent neural networks," *Design and Applications*, vol. 5, pp. 64–67, 2001.

[39] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[40] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," in *NIPS 2014 Workshop on Deep Learning*, 2014.

[41] Z. Yu, J. Lian, A. Mahmoody, G. Liu, and X. Xie, "Adaptive user modeling with long and short-term preferences for personalized recommendation." in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, 2019, pp. 4213–4219.

[42] H. Chi, H. Xu, M. Liu, Y. Bei, S. Zhou, D. Liu, and M. Zhang, "Modeling spatiotemporal periodicity and collaborative signal for local-life service recommendation," *arXiv preprint arXiv:2309.12565*, 2023.

[43] H. Sun, G. Yu, P. Zhang, B. Zhang, X. Wang, and D. Wang, "Graph based long-term and short-term interest model for click-through rate prediction," in *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 2022, pp. 1818–1826.

[44] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2014, pp. 701–710.

[45] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," in *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[46] D. P. Vassileios Balntas, Edgar Riba and K. Mikolajczyk, "Learning local feature descriptors with triplets and shallow convolutional neural networks," in *Proceedings of the British Machine Vision Conference*, 2016.

[47] U. Ruby and V. Yendapalli, "Binary cross entropy with deep learning technique for image classification," *Int. J. Adv. Trends Comput. Sci. Eng*, vol. 9, no. 10, 2020.

[48] R. Burch, F. N. Najm, P. Yang, and T. N. Trick, "A monte carlo approach for power estimation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 1, no. 1, pp. 63–71, 1993.

[49] J. Ni, J. Li, and J. McAuley, "Justifying recommendations using distantly-labeled reviews and fine-grained aspects," in *Proceedings of EMNLP-IJCNLP*, 2019, pp. 188–197.

[50] Y. Liu, W. Wei, A. Sun, and C. Miao, "Exploiting geographical neighborhood characteristics for location recommendation," in *Proceedings of the 23rd ACM International Conference on Information & Knowledge Management*, 2014, pp. 739–748.

[51] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *International Conference on Learning Representations*, 2019.

[52] Z. Hou, X. Liu, Y. Cen, Y. Dong, H. Yang, C. Wang, and J. Tang, "Graphmae: Self-supervised masked graph autoencoders," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 594–604.

[53] G. Zhou, X. Zhu, C. Song, Y. Fan, H. Zhu, X. Ma, Y. Yan, J. Jin, H. Li, and K. Gai, "Deep interest network for click-through rate prediction," in *Proceedings of the 24th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2018, pp. 1059–1068.