Liu *et al. Journal of Cloud Computing* (2023) 12:89
https://doi.org/10.1186/s13677-023-00460-4

Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH

Open Access

A large-scale data security detection method based on continuous time graph embedding framework

Zhaowei Liu[1], Weishuai Che[1], Shenqiang Wang[1], Jindong Xu[1] and Haoyu Yin[2*]

Abstract

Graph representation learning has made significant strides in various fields, including sociology and biology, in recent years. However, the majority of research has focused on static graphs, neglecting the temporality and continuity of edges in dynamic graphs. Furthermore, dynamic data are vulnerable to various security threats, such as data privacy breaches and confidentiality attacks. To tackle this issue, the present paper proposes a data security detection method based on a continuous-time graph embedding framework (CTDGE). The framework models temporal dependencies and embeds data using a graph representation learning method. A machine learning algorithm is then employed to classify and predict the embedded data to detect if it is secure or not. Experimental results show that this method performs well in data security detection, surpassing several dynamic graph embedding methods by 5% in terms of AUC metrics. Furthermore, the proposed framework outperforms other dynamic baseline methods in the node classification task of large-scale graphs containing 4321477 temporal information edges, resulting in a 10% improvement in the F1 score metric. The framework is also robust and scalable for application in various data security domains. This work is important for promoting the use of continuous-time graph embedding framework in the field of data security.

Keywords Graph representation learning, Dynamic graph, Data Security, Large-scale graph, Graph neural network, Edge computing

Introduction

Over the past few years, there has been significant growth in graph (network) data, which has been widely used in interdisciplinary fields such as social science [1], biology [2], and information science [3]. Moreover, as a subcategory of machine learning, graph data processing plays an essential role in practical applications. For example, in fields such as healthcare, processing network data can assist doctors in accurately diagnosing patients [4]. However, the use of graph data also raises significant security concerns, such as protecting sensitive information, ensuring data privacy, and preventing malicious attacks. In addition, as machine learning and artificial intelligence techniques become more prevalent in graph data analysis, the potential impact of security breaches and data manipulation is even more significant [5].

Graph representation learning involves transforming graph data into low-dimensional vector representations, associating the attributes of graph data in vector space. To achieve better performance and model accuracy, a large amount of data is usually required for training. These data often contain sensitive information such as personal privacy, trade secrets, etc. Therefore, data security is of great importance when storing and processing this data, such as protecting sensitive information, ensuring data privacy, and preventing malicious attacks. In addition, graph representation learning may face attacks against the model, leading to unexpected outputs or

*Correspondence:
Haoyu Yin
dhyy@yt.shandong.cn
[1] Department of Computer Science, Yantai University, Yantai, China
[2] Intellectual Property Protection Center, Yantai, China

leaks of sensitive information. Despite most graph representation learning methods relying on static graphs, the majority of graphs in the real world are dynamic and constantly evolving as nodes and links are added, removed, and modified. As an increasing amount of sensitive data is collected and stored in dynamic graphs, it becomes increasingly critical to ensure the security and privacy of this data. Various types of attacks, including link prediction attacks and node attribute inference attacks, can be launched to compromise the confidentiality and integrity of data in dynamic graphs. Since temporal information is crucial for accurately modeling, predicting, and understanding graph data [6], processing real-time data from dynamic graphs has emerged as a central research area in edge computing [7, 8]. Dynamic graphs can incorporate deep learning models from static graphs that disregard temporal information. However, this approach has been proven suboptimal. The limitations of dynamic graph embedding methods in network security applications are manifested in the detection of malicious network activity. Existing methods struggle to capture the complex temporal interactions between network nodes, resulting in poor performance in detecting network security threats [9]. Dynamic graph representation learning is a relatively new area of research, with some studies focusing on discrete-time dynamic graph learning, which involves a series of snapshots of graphs [10]. However, these methods may not be appropriate for real-world scenarios, such as social networks, where time is continuous and sensitive data is prevalent. Simultaneously, modeling non-linear changes in social networks using dynamic graph representation learning methods is challenging, which has implications for detecting social engineering attacks in network security [11]. Therefore, it is imperative to develop data security and privacy detection methods for continuous time graphs to ensure the protection of this sensitive data.

Recently, some methods have been proposed to support continuous-time scenarios [12]. Graph data representations contain a wealth of semantic information, and in natural language processing, skip-gram models capture some of this information by learning continuous vector representations of the relationships between words. In the field of graph embedding, skip-gram models learn node sequences generated by DeepWalk [13] and Node2vec [14], where node sequences are extracted through random walks. Therefore, based on random walks, this paper proposes a continuous time graph embedding framework that can be used to detect data security and privacy threats in continuous time graphs. This approach incorporates temporal dependencies into node embeddings for real-time prediction of data, such as in the Internet of Things (IoT) [15, 16], blockchain [17, 18], and connected vehicle networks [19]. For example, in industry, IoT devices are used to manage production parameters such as machine operating status, temperature, and pressure. These data are stored on cloud servers and can be accessed and managed through industrial control systems. If attackers can gain access to these data, they can take a series of malicious actions, such as exploiting temporary vulnerabilities in machines for illegal intrusion, disrupting or interfering with production, or selling inferior industrial machines on the market to gain economic gain. Because these data are crucial for industrial production, protecting and encrypting industrial IoT devices and data is essential [20, 21].

As network sizes continue to expand, the time attributes between nodes in constructed large-scale continuous-time graphs also increase. However, existing continuous-time graph embedding methods have limitations in effectively capturing the dynamic nature of large-scale graphs. To ensure the accuracy of prediction models and improve the security of large-scale data during processing [22], this paper proposes a continuous-time dynamic graph embedding framework (CTDGE). The framework algorithm comprises three primary steps: (1) graph partitioning, (2) continuous-time graph embedding, and (3) graph aggregation. Specifically, the CTDGE algorithm initially partitions the dynamic large-scale graph into non-overlapping subgraphs using an edge-based graph partitioning technique, which guarantees a balance of edge and weight partitioning. This partitioning approach is suitable for most graph embedding algorithms that rely on edge sampling, as it reduces computational complexity and enhances embedding quality (as discussed in "Reducing computational complexity and improving embedding quality in multi-level graph embedding"). This paper employs a random walk depth graph model that incorporates temporal dependencies into the node embeddings of subgraphs. This model improves the efficiency of dynamic graph embedding while learning from the sequential nature of subgraphs maintains efficient parallel processing. With the improvement of data security in fields such as the Internet of Things, existing continuous-time graph embedding methods are insufficient for adapting to new data processing and there are limitations to existing security detection techniques. However, the continuous-time graph embedding framework proposed in this paper can be combined with existing and future embedding methods to leverage machine learning techniques for identifying potential threats and privacy vulnerabilities in dynamic graphs. The framework can provide effective countermeasures to prevent these threats.

The main technical contributions of this paper are summarized as follows:

- This paper presents an edge-based graph partitioning algorithm suitable for large-scale continuous-time graphs, which can divide the graph into non-overlapping subgraphs.
- This paper proposes a time-respecting random wandering model to capture the continuity of data during embedding and ensure data security through node detection.
- This paper improves graph aggregation algorithms to enhance the accuracy of large-scale continuous-time graph embedding.

Related work

This section provides an overview and classification of recent graph embedding methods used for data security detection [23, 24].

Static graph embedding method

The current methods for solving static graph embedding can be classified into three categories: matrix decomposition, random walk, and deep learning [25]. The static graph embedding model based on matrix decomposition performs feature decomposition on the node association information matrix and attribute information matrix. It then fuses the decomposed attribute embedding and structural embedding to generate a low-dimensional embedded representation of nodes. While the matrix decomposition method improves embedding accuracy, it is computationally intensive and relatively expensive, particularly for large-scale data. The random walk-based static graph embedding model obtains a training corpus by conducting random walks, and then integrates the corpus into Skip-Gram to obtain low-dimensional embeddings of nodes. The most popular models of this type are DeepWalk [13]and Node2vec [14]. However, these models are limited to random walks and do not take into account the temporal properties of edges.

A graph neural network (GNN) is a deep learning model that specializes in processing graph data [26]. GNN-based static graph models aggregate the embeddings of node neighborhoods and iteratively update them, using the current embedding and the embedding of the previous iteration to generate a new representation. The GNN model captures inter-node message-passing relationships through multiple iterations, allowing the generated embeddings to characterize the global structure [27]. Graph neural networks include several models, such as graph convolutional networks [28] for neighborhood aggregation, recurrent neural networks [29] for combining with deep learning, neural networks with attention mechanisms [30], adversarial networks [31] for adversarial learning, and graph transformers [32]. The GNN model significantly enhances the embedding model's representation capability. Combining the deep model with semi-supervised techniques provides new ideas for the scalability of graph embedding [23, 33].

Embedding method for discrete-time dynamic graphs

Dynamic graph embedding methods typically incorporate a temporal dimension into static graph embedding approaches [10]. As a result, dynamic graph embedding methods can be categorized into matrix decomposition, random walk, and deep learning approaches. These methods are further divided into discrete and continuous models based on the graph's evolutionary model.

Discrete-time graph embedding involves processing time windows to learn node representations in snapshots, and is divided into two specific categories.

- Single Snapshot Model: A static model is used to create a snapshot of the graph and predict the next snapshot in the dynamic graph [34]. Another approach to implement this is TI-GCN (Time Interval Graph Convolutional Network), which uses residual structures to embed discrete-time dynamic graphs [35]. These works use information from multiple snapshots represented by the edge differences in addition to a single snapshot [36].
- Multi-Snapshot Models: For the random wandering set, each snapshot is computed separately, and they learn the final node embedding together [37]. Recurrent neural networks (RNNs) are used to process serial data, such as graphs [29]. Recently, GANs have been combined with RNNs instead of using node features as inputs to RNNs [38]. DACHA [39] introduces a dual convolutional network to capture the impact of entity and historical relationships and uses a self-attentive encoder to model temporal dependencies in the knowledge graph. STGCNs extend graph convolution into temporal and spatial graph convolution networks to capture temporal changes in dynamic graphs, particularly to model dynamic parameters in snapshots of adjacent graphs [40].

Many of the current discrete-time graph embedding methods require manual selection of time windows, and as a result, they lose the order of edge formation, reflecting only a portion of the graph information. Additionally, large-scale discrete-time graph representations can be inefficient in memory usage and impractical to apply, as noted by Cui et al. [41].

Embedding method for continuous-time dynamic graphs

Recently, there has been increasing attention on dynamic graph embeddings that consider edges with continuous

time properties [42]. For instance, this continuous data can enable travel businesses to intelligently predict users' interests and preferences, offer them scientifically designed travel routes, and boost their revenue [43]. A temporal-based random walk model has been proposed that directly incorporates temporal dependencies into the sequence of nodes generated by the random walk. Attention-based mechanisms have also been developed to learn the importance of temporal random walks between nodes and their neighbors [44]. Some methods capture the evolution of graph structures through temporal random walks, resulting in embeddings that are more specific [6, 45].

When nodes or edges are added or deleted in the graph, the associated nodes' embeddings are updated by aggregating information from their new neighbors. This class of methods is referred to as local neighborhood models. In DyGCN [41] and TDGNN [46], the authors extend the GCN-based approach by incorporating temporal and spatial information to generalize embeddings for efficient dynamic graph representation learning, while incorporating adaptive mechanisms in the model.

Some methods address the information asymmetry problem in graphs by assigning priority to nodes [47]. However, the core idea of continuous temporal graph embedding is to extend existing models with special storage modules designed for node classification where labels are fixed over time, making them unsuitable for general frameworks [48, 49]. Independent module-based frameworks have shown promising results in industrial applications.

In the past, the main disadvantage of using vertex-based partitioning was the uncertainty surrounding the degree of each vertex, which made it difficult to achieve a balanced partitioning of the graph. However, the latest approach, which is based on edges, simplifies the partitioning process and ensures a more even distribution of the graph [50]. Given the potential of continuous-time graph embedding models and the necessity of dealing with large-scale graphs [6], this paper aims to implement a framework for large-scale continuous-time graph embedding.

Framework

To expedite the processing of large-scale dynamic graphs, this paper presents a framework comprising three components, as depicted in Fig. 1: (1) Graph Partitioning; (2) Continuous-Time Graph Embedding; and (3) Graph Aggregation.

Definitions

Dynamic graphs can be categorized as discrete-time graphs and continuous-time graphs, depending on how time is represented. A discrete-time graph consists of a sequence of static graphs, each representing a specific time interval, denoted as $\mathcal{G} = \{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_T\}$. On the other hand, a continuous-time graph is defined as $G = (V, E_T, \mathcal{T})$, where E_T is the set of edges between vertices V with temporal properties, and $\mathcal{T} : E \rightarrow \mathbb{R}^+$ maps each edge to a non-negative real number, representing the time at which the edge occurs. This mapping is represented here as a temporal function. In a continuous-time graph, each edge $e = (u, v, t) \in E_T$ has a unique timestamp $t \in \mathbb{R}^+$.

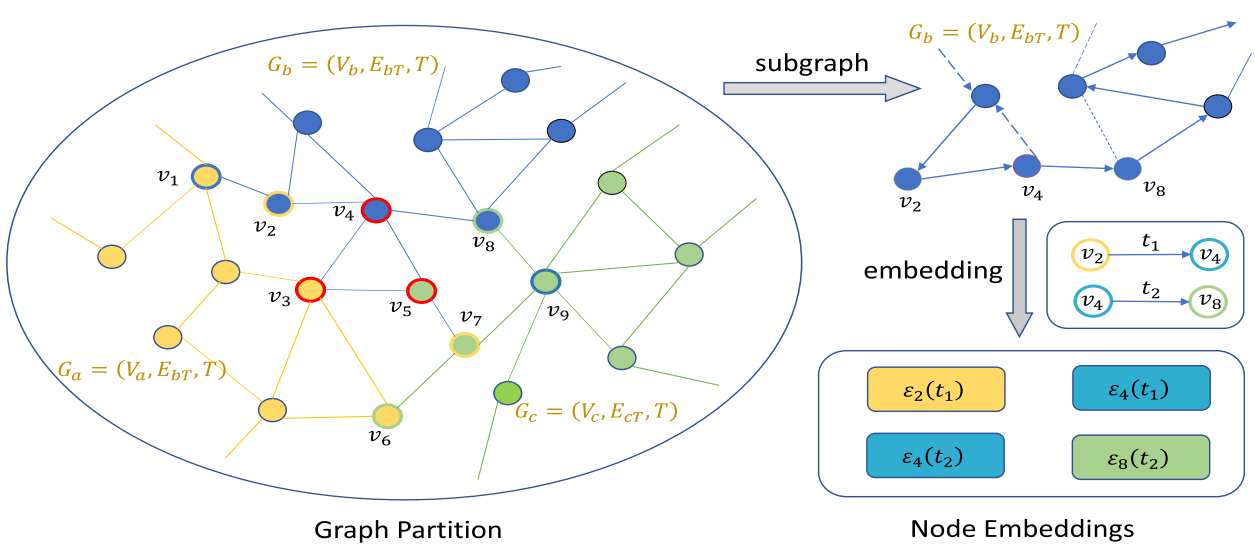

Fig. 1 A dynamic graph embedding framework based on large-scale continuous time

Graph partition

Due to the memory and runtime demands of big data, large-scale continuous-time graph embedding poses a significant challenge. Currently, the most efficient approach involves partitioning the graph into multiple clusters for embedding. Graph partitioning methods generally fall into two categories: vertex-based and edge-based partitioning. While the vertex-based method is straightforward, it cannot ensure a balanced division of the graph. On the other hand, the edge-based method can achieve a balanced division, but it may not preserve the temporal continuity. To address this issue, our paper proposes a temporal attribute-based edge delineation method with the following key features.

- This paper proposes an edge-based graph partitioning algorithm that improves the division of unbalanced graphs. Unlike vertex-based methods, the time complexity of graph embedding is determined by the number of edges in the subgraph rather than the number of vertices. Consequently, the proposed algorithm is more likely to reduce running time.
- Moreover, the algorithm partitions the graph based on the temporal properties of its edges, while preserving the similarity between vertices to the maximum extent possible.

For a given dynamic large-scale graph $G = (V, E_T, \mathcal{T})$, all edges are divided into k distinct subgraphs $G_k = \left(V_k, E_{k_T}, \mathcal{T}\right)$ without overlapping, i.e.,

$$E = \cup_{k=1}^{K} E_k \quad \forall i, j : i \neq j \Rightarrow E_i \cap E_j = \emptyset. \tag{1}$$

The variable k represents a predetermined number of subgraphs. It's important to note that subgraphs resulting from graph partitioning can have overlapping vertices, whereas in continuous-time dynamic graphs, the order of vertices is significant. The formula $N_{ij} = \left\{v_k \mid v_k \in V_i \cap V_j\right\}$ denotes the set of overlapping vertices in the subgraph between G_i and G_j, where V_i and V_j represent the vertex sets of G_i and G_j, respectively.

The graph partition section of Fig. 1 displays three subgraphs denoted as G_a, G_b, and G_c, with their edges highlighted in yellow, blue, and green respectively. Observation reveals that v_1 is connected by both yellow and blue edges, hence v_1 belongs to G_a and G_b (i.e., $v_1 \in N_{ab}$). Similarly, this paper obtain $N_{ab} = \{v_1, v_2, v_3, v_4, v_5\}$, $N_{bc} = \{v_3, v_4, v_5, v_8, v_9\}$, and $N_{ac} = \{v_3, v_4, v_5, v_6, v_7\}$. Moreover, vertices v_3, v_4 and v_5 are common to all three subgraphs.

To ensure effective application of graph partitioning algorithms on large-scale graphs, it is crucial to maintain a relatively consistent number of edges within each subgraph, ideally as close to $|E|\ /K$ as possible. One common metric for evaluating the balance of the partitioning is the standard deviation of subgraph sizes.

$$std = \sqrt{\frac{\sum_{k=1}^{K}\left(\frac{\left|E_{k_T}\right|}{|E|/K} - 1\right)^2}{K}}. \tag{2}$$

For a subgraph G_k, the set of d-dimensional embedding vectors of the subgraph embedding and the global embedding is $\boldsymbol{Z}^{(k)}, \boldsymbol{Y}^{(k)}, \in \mathbb{R}^{|V_k| \times d}$. The objective function of graph partitioning optimization is:

$$\min \sum \left\|\boldsymbol{Z}^{(k)} - \boldsymbol{Y}^{(k)}\right\|^2. \tag{3}$$

Moreover, the study necessitates an examination of the communication among all partitions. This scrutiny is well illustrated by the graph aggregation process in the framework, which defines the overlapping vertices in the graph partition.

$$\min \sum_i \sum_i \left|N_{ij}\right|, \tag{4}$$

where N_{ij} is the set of overlapping vertices between two continuous-time subgraphs G_i and G_j, as discussed in Eq. (2), $\forall i, \forall j \in [1, K], i \neq j$

Recent studies have introduced effective partitioning methods [51]that yield good results. However, the initial time in a continuous-time system can impact the size of the partition. Moreover, in a continuous-time system $\mathbb{T} = \mathbb{R}^+$, the order in which nodes are connected by edges is crucial. It is important to note that the weights of the continuous-time subgraph G_k are determined by the corresponding time $t_* = \mathcal{T}(e_i)$. In order to tackle this issue, an edge partitioning algorithm based on continuous-time graphs is proposed, which takes into consideration the correlation time. Furthermore, to maintain the structural similarity of each subgraph to the original graph, while minimizing the effect of total weights on training, the weight balance of graph partitions is taken into account. The procedure for implementing this algorithm is outlined as follows:

- First, all subgraphs (K in total) are initially open to accept new edges. Once the subgraph reaches its capacity (i.e., the maximum number of allowed edges), it is considered closed.
- Second, after partitioning the graph, an edge $e = (u, v)$ will be assigned to the subgraph that contains the vertex u or v with the lowest weight, while ensuring that the total weight of all subgraphs remains conserved.

- Finally, to ensure balance in the number of edges $|E_{k_T}|$ within each subgraph $k = \{1, 2, \cdots K\}$ over time, a threshold t_e is established. If the difference between the maximum and minimum number of edges in subgraphs $\left(\max\left(\left|E_{k_T}\right|\right) - \min\left(\left|E_{k_T}\right|\right)\right)$ is greater than or equal to t_e, the next incoming edge with the smallest weight will be assigned to the subgraph with the smallest $|E_{k_T}|$ as time progresses (from the beginning to the current state).

Continuous-time graph embeddings

In the dynamic embedding process, this paper defines each dynamic subgraph as $G_k = \left(V_k, E_{k_T}, \mathcal{T}\right)$, where V_k is a set of vertices and E_{k_T} is the set of edges between vertices in V_k. The set E_{k_T} is the set of edges that occur consecutively between the subgraph vertices in V_k, and $\mathcal{T} : E_k \to \mathbb{R}^+$ is a time function that assigns a timestamp to each edge in the subgraph. For the optimal partitioned subgraph, each edge $e_i = (u, v, t) \in E_{k_T}$ can be assigned a specific timestamp $\mathbb{T} \in \mathbb{R}^+$.

In a continuous-time subgraph, the set $\mathcal{T} \subseteq T$ represents the time span during which information on an edge occurs, where T is the time domain. The continuous-time system is defined as $\mathbb{T} = \mathbb{R}^+$. In such a graph, a valid walk is a sequence of consecutive nodes that have temporal properties themselves and are connected by edges between nodes with non-decreasing temporal information. Specifically, the timestamp of each edge captures the contact time between two nodes, so that a valid time walk represents feasible routes that respect temporal information.

For a valid random walk from vertex V_1 to V_k in G_k, a sequence of vertices $\langle v_1, v_2, \cdots, v_k\rangle$ is valid if $\langle v_i, v_{i+1}\rangle \in E_{k_T}$ for $1 \le i < k$, and $\mathcal{T}(v_i, v_{i+1}) \le \mathcal{T}(v_{i+1}, v_{i+2})$ for $1 \le i < (k-1)$. If there exists a time walk from vertex u to vertex v for any arbitrary vertices $u, v \in V_k$, then u is time-connected to v.

This paper defines the embedding problem for continuous-time subgraphs as follows: Given a continuous-time graph $G = (V, E_T, \mathcal{T})$, the goal is to map its nodes to a D-dimensional vector space and learn a time-varying feature representation function $f : V \to \mathbb{R}^D$. This approach is suitable for link prediction of temporal attributes and other machine learning tasks. The first step of graph embedding involves determining the node at which the random walk starts. In this paper, the starting time t_* is drawn from a uniform weighted distribution $\mathbb{F}_S$, and the closest edge E to time t_* is found. Alternatively, the initial edge $e_i = (v, w)$ and its associated time $t_* = \mathcal{T}(e_i)$ can be drawn from an arbitrary (uniform or weighted) distribution $\mathbb{F}_S$. To achieve dynamic graph embedding, this paper employs time-varying embeddings, which distinguishes our framework from existing approaches that use random wandering on static graphs. Strategies for selecting initial time edges that are temporally biased or unbiased are discussed in [6]. Specifically, the proposed method starts directly from the initial time edge of each subgraph.

In each subgraph, the time random walk is initiated with the selection of the initial edge $e_i = (u, v, t)$, and this paper defines the set $\Gamma_t(v)$ of temporal neighbors of node v at time t as follows.

$$\Gamma_t(v) = \left\{\left(w, t'\right) \mid e = \left(v, w, t'\right) \in E_{k_T} \wedge \mathcal{T}(e) > t\right\} \tag{5}$$

It should be noted that a node w may appear multiple times in the temporal neighborhood $\Gamma_t(v)$ of a node v, due to the presence of multiple edges with distinct timestamps between the two nodes. This paper focuses solely on unbiased time, and favors the selection of temporal neighbors in the second distribution Γ_t. Specifically, this paper bias the sampling strategy towards walks that exhibit smaller "in-between" times on consecutive edges. This way, the subgraph embedding considers each pair of consecutive edges (u, v, t) and $(v, w, t+k)$ encountered by the random walk. For example, if k is small, the random walk sequence $(v_2, v_4, t), (v_4, v_9, t+k)$ can be sampled. Since v_4 is linked to v_2 and v_9, respectively, it is likely that v_2 and v_9 are also connected. On the other hand, if k is large, this sequence is unlikely to be sampled. Consequently, if v_4 interacts with v_2 and v_9 at very different times, they are more likely to be separated and unaware of each other's existence.

Given a time walk $\mathcal{S}_t$, the task of learning node embeddings in continuous-time dynamic graphs is formulated as an optimization problem.

$$\max_f \log \Pr(\mathrm{W_T} = \{\mathrm{v_{i-\omega}}, \cdots, \mathrm{v_{i+\omega}}\} \backslash \mathrm{v_i} \mid \mathrm{f(v_i)}) \tag{6}$$

The node embedding function $f : V \to \mathbb{R}^D$ is used to optimize the context window size, denoted by ω. The window W_T is defined as $\mathcal{T}(v_{i-\omega}, v_{i-\omega+1}) < \cdots < \mathcal{T}(v_{i+\omega-1}, v_{i+\omega})$ and is a subset of the time walk $\mathcal{S}t$. It is assumed in this paper that when the source node v i is observed, there exists conditional independence between the nodes within the temporal background window W_T.

$$\Pr(\mathrm{W_T} \mid \mathrm{f(v_i)}) = \prod_{\mathrm{v_{i+k}} \in \mathrm{W_T}} \Pr\left(\mathrm{v_{i+k}} \mid \mathrm{f(v_i)}\right) \tag{7}$$

This paper utilizes a graph partitioning algorithm to divide a continuous-time dynamic graph $G = (V, E_T, \mathcal{T})$ into k subgraphs. The random walk space for each subgraph G_k is represented by $\mathbb{S}$. The space of temporal random walks of subgraph G_k is denoted as $\mathbb{S}_T$, representing only the subset of random walks that respect time. To

ensure temporal coherence, this paper defines the context window size of jump frames as the minimum length for each temporal walk. Specifically, a set of temporal random walks $\{\mathcal{S}t_1, \mathcal{S}t_2, \cdots, \mathcal{S}t_k\}$ is used to obtain a series of context windows of size ω, and the total number of these temporal context windows is denoted as β.

$$\beta = \sum_{i=1}^{k} \left|\mathcal{S}_{t_i}\right| - \omega + 1 \tag{8}$$

When sampling a set of time wanderings, β is usually set to a multiple of $N = |V|$.

This paper provides an overview of the process of learning continuous-time subgraph embeddings using Alg. 1 and the Temporal Random Walk (TRW) [6]. Additionally, the subgraph embedding framework presented in this paper can be utilized in a depth-model based approach, as the temporal random walk can serve as an input vector to the neural network.

Require: a (un)weighted and (un)directed continuous time subgraphs $G_k = (V_k, E_{k_T}, \mathcal{T})$, embedding dimensions D, context window size ω, temporal context window count β
Ensure: the dynamic node embedding matrix Z
  Set the maximum length of time walk $L = 50$
  Initialize the set of time walks $\mathcal{S}_T$ to 0
  Initialize the number of background windows $C = 0$
  Use G_k to precompute the sampling distribution $\mathbb{F}_S$
  $G_k = (V_k, E_{k_T}, \mathcal{T}, \mathbb{F}_S)$
  While $\beta - C > 0$ **do** (Eq. 8)
    Sample an edge $e_* = (v, u)$ via distribution $\mathbb{F}_S$
    $t_* = \mathcal{T}(e_i)$
    $S_t = (G', e_* = (v, u), t, L, \omega + \beta - C - 1)$
    According to Eq. (5)
    if $|S_t| > \omega$ **then**
      Add the temporal walk S_t to $\mathcal{S}_T$
      $C = C + (|S_t| - \omega + 1)$
  end while
  $\boldsymbol{Z} = (\omega, D, \mathcal{S}_T)$
  return the dynamic node embedding matrix $\boldsymbol{Z}$

Algorithm 1 Continuous-Time Subgraph EmbeddingsGraph aggregation

In the final stage of CTDGE, distributed subgraph embeddings are aggregated. However, the effective walks in a subgraph embedding, which are represented by a sequence of nodes connected by edges with non-decreasing timestamps, present a challenge to graph aggregation due to uncertain continuity between subgraphs. To address this issue, a basic idea of global aggregation is employed in this study to identify a global vector space in time that can map multiple local subgraph embedding spaces. The subspaces are then mapped using the overlapping vertex set N.

Assuming that a vertex v_m belongs to multiple subgraphs, such as G_i and G_j that are represented by $v_m \in N_{ij}$, the local embedding vector spaces of G_i and G_j can be denoted as $\boldsymbol{Z}^{(i)} = F(G_i)$ and $\boldsymbol{Z}^{(j)} = F\left(G_j\right)$, respectively. As a result, the local embedding vectors of v_m in the subgraphs G_i and G_j can be represented by $\boldsymbol{z}_m^{(i)}$ and $\boldsymbol{z}_m^{(j)}$, respectively. If a mapping function $h\left(\boldsymbol{z}_m^{(i)}, \boldsymbol{z}_m^{(j)}\right) \longrightarrow \boldsymbol{y}_m$ exists for overlapping vertices v_m, this function maps the entire subspaces $\boldsymbol{Z}^{(i)}$ and $\boldsymbol{Z}^{(j)}$ to a global vector space $\boldsymbol{Y}$.

$$h\left(\boldsymbol{Z}^{(i)}, \boldsymbol{Z}^{(j)}\right) \overset{h\left(\boldsymbol{z}_m^{(i)}, \boldsymbol{z}_m^{(j)}\right)}{\longrightarrow} \boldsymbol{Y} \tag{9}$$

This unsupervised graph global aggregation algorithm is designed to be low-complexity. It is both simple and efficient, and involves normalization and combination processes. Specifically, the algorithm first identifies the set of overlapping vertices in all subgraphs, denoted by $N_{all} = V_1 \cap V_2 \cap \ldots \cap V_K$. For each vertex $v_m \in N_{all}$, the local embedding vector $\boldsymbol{z}_m^i = \left[z_m^i(1), z_m^i(2), \ldots, z_m^i(d)\right]$ in each cluster i is then normalized, with

$$\boldsymbol{z}_m^{(i)'} = \left[\frac{z_m^i(1) - e_m^i}{\sigma_m^{(i)}}, \frac{z_m^i(2) - e_m^i}{\sigma_m^{(i)}}, \ldots, \frac{z_m^i(d) - e_m^i}{\sigma_m^{(i)}}\right]. \tag{10}$$

The mean of $\sum z_m^i$ is calculated as

$$e_m^i = \frac{\sum z_m^i(1) + z_m^i(2) + \cdots + z_m^i(d)}{d}, \tag{11}$$

and the variance is denoted as ${\sigma_m^{(i)}}^2$.

After normalization, the normalized embedding vectors are used to combine the global space, and overlapping vertices are generated through different local clustering techniques. The average value of the normalized vector for vertex v_m can be obtained using Eq. (10).

$$\boldsymbol{z}'_m = \frac{\sum_{i=1}^{K} \boldsymbol{z}_m^{(i)'}}{K} \tag{12}$$

The mapping function for vertex v_m can be represented as $z_m^i = h_m\left(\boldsymbol{z}_m^{(i)}, \boldsymbol{z}_m^{(j)}\right)$. To calculate the global standard embedding vector, this paper takes the average value of the transformation embeddings of all vertices m in the set N_{all}, denoted as:

$$\boldsymbol{z}^{(all)} = \frac{\sum_{m \in N_{all}} \boldsymbol{z}'_m}{|N_{all}|}. \tag{13}$$

Additionally, the subgraph's vector space $\boldsymbol{Z}^{(i)}$ is mapped to the global vector space $\boldsymbol{y}_i$, denoted as:

$$\boldsymbol{y}i' = \boldsymbol{Z}^{(i)} - \mathrm{dist}^{(\mathrm{i})}, \tag{14}$$

where $\mathrm{dist}^{(\mathrm{i})} = \frac{\sum_{\mathrm{v_m} \in \mathrm{V_i} \cap \mathrm{N_{all}}} \left(\boldsymbol{z}_{\mathrm{m}}^{(\mathrm{i})} - \boldsymbol{z}^{(\mathrm{all})}\right)}{|\mathrm{V_i} \cap \mathrm{N_{all}}|} \quad \forall \mathrm{v_m} \in \mathrm{V_i} \cap \mathrm{N_{all}}$. Finally, the global feature $\boldsymbol{Y}$ of the graph aggregation is denoted as $\boldsymbol{Y} = \left[\boldsymbol{y}'_1, \boldsymbol{y}'_2, \ldots, \boldsymbol{y}'_K\right]$.

Experiments

The experiment aims to investigate the effectiveness of a proposed continuous-time dynamic graph embedding (CTDGE) framework, which uses temporal graphs with

different temporal characteristics and number of scales. This paper evaluate the performance of CTDGE in data security using metrics such as link prediction and node classification tests. Additionally, the performance of the framework is compared with current mainstream methods as the graph size is increased. The experimental results show that the proposed framework achieves better results for large-scale continuous time graphs while significantly reducing the running time.

Datasets

To compare the performance of various graph embedding methods, the dataset needs to fulfill certain criteria, such as selecting continuous-time dynamic graphs with timestamps from $\mathbb{T} = \mathbb{R}^+$. This paper examines and analyzes two types of datasets. The first type comprises publicly available continuous-time datasets. This study collects and pre-processes four datasets, as shown in Table 1. The second type is a large-scale dynamic dataset obtained from the web to verify the embedding on dynamic bigraphs. This paper uses the Yelp and Tmall datasets, which track a large number of internet users' reviews of merchants (e.g., restaurants and shopping malls) and user information on Tmall products, respectively, as shown in Table 2.

Table 2 Statistics of large scale dynamic graphs

Dataset	nodes	edges	time steps
Yelp	42 653	834 291	17
Tmall	36 183	4 321 477	10

Experimental setup

The paper employs continuous-time dynamic graphs to model the learning framework and compares different baseline methods from various categories. Two vector-based graph representation learning approaches, namely DeepWalk and Node2vec, are used as typical examples of link prediction using static random wandering. Graph Convolutional Networks (GCN) and Graph Attention Networks (GAT) are considered as static networks, and the graph is assumed to be static during the experiments. The paper focuses on the node detection task for each graph, and aligns these embedding methods to the same vector space. To assess the framework's performance comprehensively, the paper compares it with two continuous-time graph embedding methods (CTDNE and TempGAN) and one discrete-time graph embedding method (DynGEM). The experimental procedure follows the hyperparameter settings suggested by the continuous-time uniformity [6] to enable better comparison with dynamic graph embedding methods.

- DeepWalk is a method for mining graph structure data based on random walks inspired by the skip-gram model. In this paper, to enable comparison of experiments, this paper set three hyperparameters to the default values ($D = 128, r = 10, ns = 10$), and leave the other two hyperparameters to vary among several values: $L \in \{40, 60, 80\}$ and $cs \in \{6, 8, 10\}$.
- Node2vec [14]. Node2vec is a graph embedding method with multi-neighborhood sets and preserving the higher order similarity of nodes. To better present the experimental data, node2vec introduces new hyperparameters for the grid search, putting $p, q \in \{0.25, 0.50, 1, 2\}$.
- GCN [52]. GCN is a multilayer network, where each convolutional layer handles first-order neighborhood information, and multi-order neighborhood information transfer is achieved by overlaying multiple convolutional layers.
- GAT [30]. GAT is an algorithm for increasing attention mechanism on GCN which uses a parameter matrix to learn the relative importance between nodes i and neighbors j. It enables the graph neural network to focus more on the important nodes to reduce the impact from edge noise.
- CTDNE [6]. This is a DeepWalk-based continuous-time network embedding method that captures temporal information through chronological random walks. This method is also the base embedding

Table 1 Statistics of a Dynamic Graph. $|V|$ represents the number of nodes in the graph, $|E_T|$ represents the number of temporal edges, d denotes the average node degree across all timesteps, d_{max} represents the maximum node degree across all timesteps, $\mathcal{S}_t$ represents the entire time span in days, T denotes the number of time steps in the training data

Database	$\|V\|$	$\|E_T\|$	d	d_{max}	$\mathcal{S}_t$	T
IA-EMAIL-EU	927	323.3K	15.8	232	784.87	44
FB-FORUM	899	33.7K	9.9	109	147.52	10
SOC-SIGN-BITCOINA	3.7K	24.1K	5.9	597	1791.32	11
SOC-WIKI-ELEC	7.1K	107K	12.2	602	1263.47	20

method for subgraphs. In this paper, F_s is set as a linear distribution and F_t is set as an unbiased distribution.

- TempGAN [53]. This is a method that preserves the temporal proximity between network nodes and learns representations from a temporal network in continuous time. In addition, link prediction experiments are performed using TempGAN autoencoder to evaluate the quality of the generated embeddings.
- DynGEM [34]. This is a deep autoencoder-based model that generates nonlinear embedding vectors by initializing the previous graph to improve dynamic graph embedding efficiency. This article refers to the suggested fixed parameters in [34].
- DynamicTriad [54]. DynamicTriad is a dynamic representation learning method that focuses on triples of vertices and shared neighbor nodes. For large-scale graphs, this paper refers to DynamicTriad's file tests.

Link prediction

This paper evaluates the quality of continuous-time dynamic graph representation learning in the CTDGE framework through link prediction. When generating training and test data, the suggestion in [6] is followed, which involves sorting all temporal edges in ascending chronological order. First, the process of parameter tuning is discussed to optimize the system's performance. The static graph embedding methods (Deepwalk, Node2vec, GCN, and GAT) learn the entire training data as a graph, while specific information about the time nodes is set in Table 1. In the dynamic graph embedding method, the TempGAN encoder generates node embeddings that hide 15% of the temporal links in the original graph. The $L2$ distance of the current time period is used to compute the similarity of two nodes with the same test parameters as CTDNE. The aim is to predict whether there is an edge between these two nodes in the next time period. The performance of AUC is evaluated based on the logistic regression model and 5-fold cross-validation.

This paper compares the performance of CTDGE with four static graph embedding methods (Deepwalk, Node2vec, GCN, and GAT) and three dynamic graph embedding methods (CEDNE, TempGAN, and DynGEM). Table 3 illustrates the AUC comparison. It is evident that for the LA-EMAIL-EU dataset, the proposed framework outperforms Deepwalk, Node2vec, GCN, GAT, and DynGEM by 11.1%, 4.0%, 11.8%, 3.4%, and 3.5%, respectively. Similarly, for the FB-FORUM dataset, the AUC improves by 25.5%, 6.4%, 24.7%, 4.2%, and 2.3% compared to the baseline. The framework achieves comparable performance to dynamically advanced methods (CTDNE and TempGAN) in graphs with less than 1000 nodes. For graphs with more than 1000 nodes, on the SOC-SIGN-BITCOINA dataset, the proposed framework achieves 11.9%, 9.3%, 10.9%, 7.8%, 4.9%, 1.2%, and 3.5% performance improvement. Similarly, on the SOC-WIKI-ELEC dataset, the AUC improves by 6.1%, 3.5%, 4.4%, 2.9%, 1.3%, 0.5%, and 2.7% compared to all methods. Overall, the proposed method outperforms other methods in the case of a large number of nodes and edges, demonstrating that delineating dynamic large-scale graphs and incorporating temporal dependencies in graphs are essential for learning appropriate graph representations. Finally, the framework of this paper can be combined with and generalized to other random walk and continuous time graph embedding based methods, which are important for future application studies.

Additionally, Fig. 2 presents the AUC scores for each time step of all evaluated methods. Based on these results, static methods such as Deepwalk and Node2vec perform worse than most dynamic methods but outperform some dynamic graph embedding methods such as DynGEM in large-scale datasets. This is due to the inability of dynamic learning methods to focus on the temporal properties of the dataset. Moreover, our method shows better performance than some continuous-time methods (CTDNE) on large-scale datasets (Yelp and Tmall). One possible explanation is that the global graph is partitioned by our framework to capture the graph evolution between different time steps in the subgraphs. Lastly, the temporal dimension is varied multiple times in our framework to effectively capture the temporal evolutionary characteristics of the nodes.

Table 3 AUC scores for Temporal Link Prediction

Dataset	DeepWalk	Node2vec	GCN	GAT	CTDNE	TempGAN	DynGEM	CTDGE
IA-EMAIL-EU	0.793	0.847	0.788	0.852	0.885	0.887	0.851	0.881
FB-FORUM	0.647	0.763	0.651	0.779	0.815	0.819	0.794	0.812
SOC-SIGN-BITCOINA	0.816	0.835	0.823	0.847	0.870	0.902	0.882	**0.913**
SOC-WIKI-ELEC	0.810	0.831	0.824	0.836	0.849	0.856	0.837	**0.860**

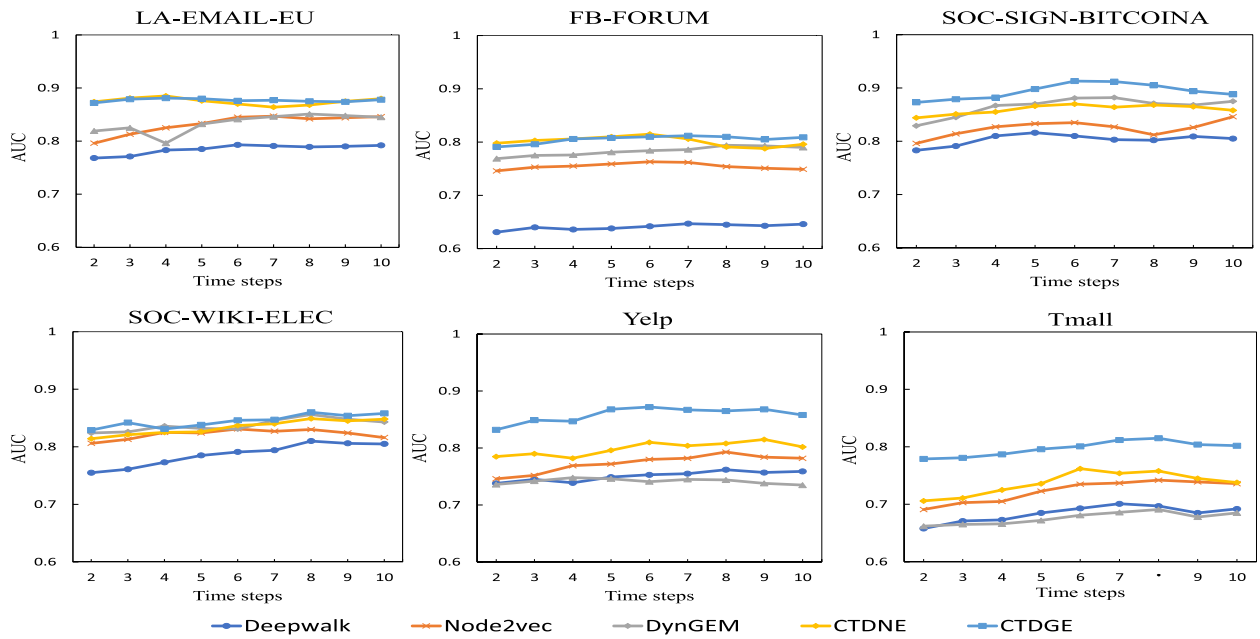

Fig. 2 Evaluation results for link prediction of dynamic datasets calculated with AUC

Large-scale dynamic graph embedding

To further demonstrate the advantages of the graph embedding framework for large-scale continuous-time graphs proposed in this paper, experiments were conducted on two dynamic graphs (Yelp and Tmall). As training on large-scale dynamic graphs requires global traversal, this was a test of computational efficiency. The four basic embedding methods studied in this paper required significant training time, and even with improved computing power, the results were unsatisfactory. In contrast, the CTDGE framework divides large-scale graphs into multiple subgraphs using graph partitioning methods, performs local random walks in each subgraph, and then aggregates the graphs to make embedding more feasible for large-scale datasets.

As shown in Fig. 3, the CTDGE framework significantly improves the performance of large-scale continuous-time graph embedding while also reducing memory usage. On the Yelp dataset, the proposed framework achieved a 13.3%, 9.3%, 6.8%, 1.2%, and 2.2% performance improvement over Deepwalk, Node2vec, CTDNE, DynamicTriad, and TempGAN, respectively. Similarly, on the Tmall dataset, CTDGE improved AUC by 15.6%, 10.6%, 7.5%, 1.5%, and 1.9% compared to Deepwalk, Node2vec, CTDNE, DynamicTriad, and TempGAN. Overall, the framework achieved an AUC gain of 6.6% across all embedding methods.

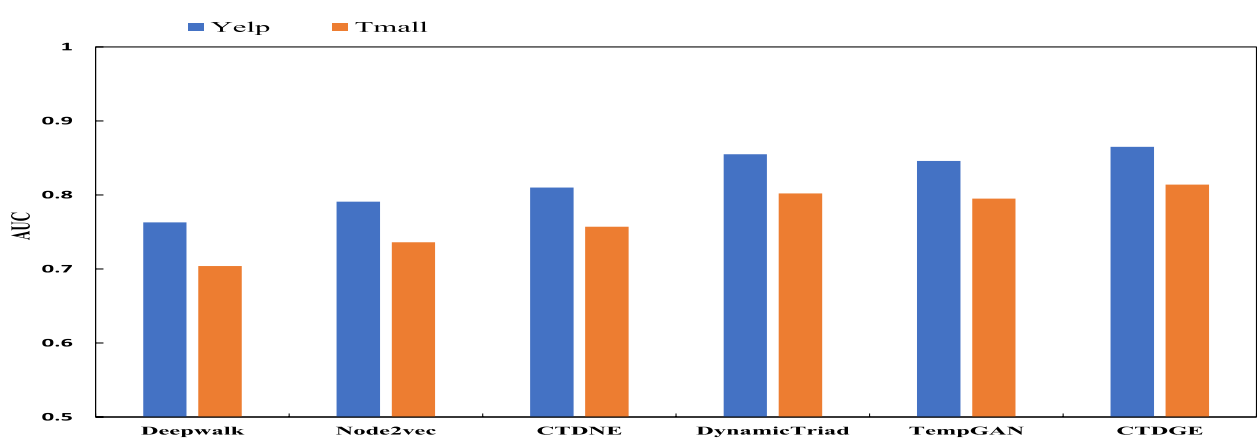

Fig. 3 Results of link prediction using various graph embedding methods for large-scale dynamic graphs

CTDGE performs global segmentation of large-scale dynamic graphs and adheres to the time sequence within subgraphs, giving a higher weight to edges that appear later in time. Experimental results consistently demonstrate that this method outperforms both DeepWalk and Node2vec. It is noteworthy that embedding methods for large-scale dynamic graphs are scarce due to the considerable computing and memory requirements. Moreover, for applications involving such graphs, our proposed framework can be combined with state-of-the-art random-walk-based methods, making it even more scalable.

Node classification

In order to evaluate the performance of large-scale graphs (with over 10,000 nodes), this paper also conducted tests on the node classification task. Table 4 displays the experimental results, showing the node classification performance of various embedding methods on large-scale graphs. The results demonstrate that CTDGE outperforms other dynamic baseline methods, achieving up to a 22.7% improvement in F1-score for node classification on large-scale graphs. This suggests that learning the representation jointly across all time steps enhances the overall performance, since it enforces continuous subgraph embeddings over time. In contrast, our framework performs global aggregation at the end, capturing the global temporal structure better than the local structure, since the node classification task considers the overall position of the embeddings. Therefore, the framework presented in this paper is better suited for large-scale data in detecting and capturing the evolution of information.

The subgraph embedding process involves an important hyperparameter, the latent space dimension. The embedding algorithm offers benefits in terms of coding efficiency and inference performance. Therefore, comparing the proposed framework with the baseline methods is carried out to evaluate the impact of embedding dimension on node classification tasks. Experimental results depicted in Fig. 4 show that CTDGE demonstrates superior embedding effectiveness on large-scale graphs compared to other methods. Additionally, CTDGE's performance stabilizes as the embedding dimension increases, starting from d=6.

Furthermore, there is a saturation point in the embedding dimension where only 20 features are sufficient to represent the node neighborhood. This value is related to the potential dimensionality of the continuous-time graph, which captures all the structural information in the graph and depends on the nature, rather than the size, of the input data. In other words, the maximum proportion of information encoded in a dimension does not depend on the number of samples.

Additionally, Fig. 5 presents the F1 scores for each time step in the large-scale dataset. The experimental results demonstrate that our framework outperforms other static and dynamic methods in node classification tasks for large-scale graphs. Finally, the base embedding component of CTDGE can be used as a modular component that can be combined with existing and future graph embedding methods.

Conclusion

This paper presents a general framework for incorporating temporal information into large-scale graph embeddings. The CTDGE framework improves the efficiency of large-scale data security detection through balanced subgraph partitioning. Additionally, the model dynamically embeds continuous-time subgraphs and captures temporal attributes in the network, which is of paramount importance for network security in the real world. The experimental results indicate that CTDGE achieved high

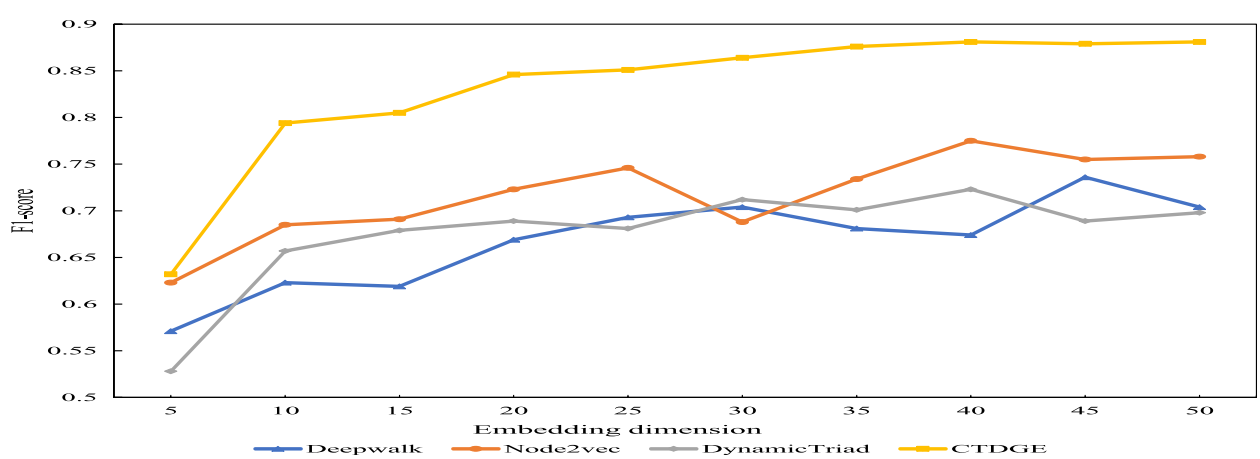

Fig. 4 Impact of embedding dimension on node classification task

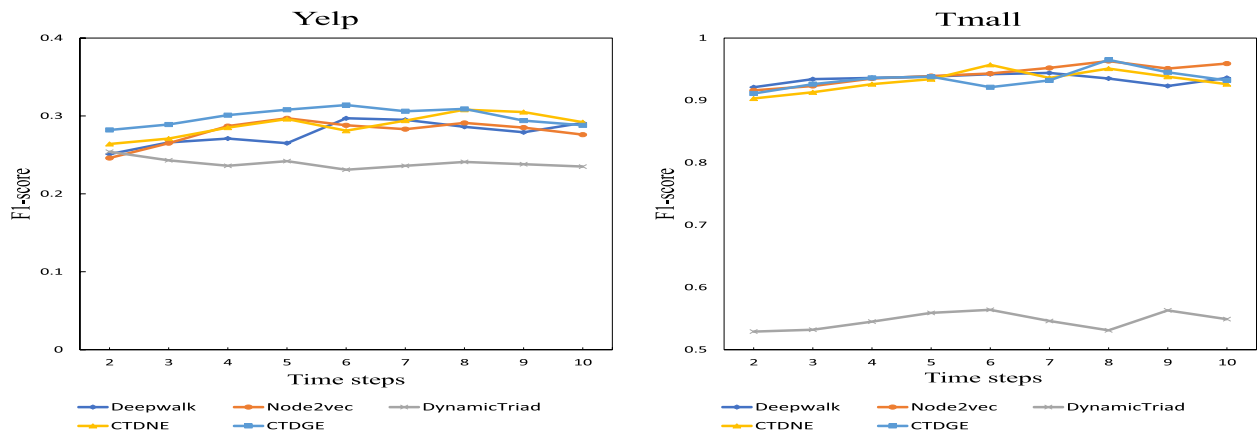

Fig. 5 Evaluation results of dynamic large-scale datasets calculated with F1-score

Table 4 Node classification model scores for large scale dynamic graphs

Dataset	DeepWalk	Node2vec	DynamicTriad	CTDGE
Yelp	0.29743	0.29735	0.25463	**0.31405**
Tmall	0.94382	0.96327	0.56271	**0.96518**

scores in link prediction and node classification tasks for large-scale data. As the dataset increases, the accuracy remains constant while the execution time decreases significantly, proving the effectiveness of the model in large-scale data security. Moreover, in real-world network testing, the model can accurately classify malicious nodes. In summary, the proposed framework achieved an average gain of 10.3% compared to embedding methods in a comprehensive analysis.

In future research, we will continue to investigate dynamic graph representation learning methods and apply them to fields such as the industrial Internet of Things to enhance data security.

Acknowledgements
The authors thank the reviewers for their insightful comments and suggestions to improve the quality of the paper. Thanks to Dr. Zhaowei Liu of Yantai University for his help in our work.

Authors' contributions
Zhaowei Liu, Weishuai Che, and Shenqiang Wang wrote the main manuscript text, Jindong Xu drew the experimental diagrams in the manuscript, Haoyu Yin drew the experimental tables in the manuscript, and all authors reviewed the manuscript. The author(s) read and approved the final manuscript.

Authors' information
Zhaowei Liu received the Ph.D. degree from the Shandong University, Jinan, in 2018. Currently, he is a Professor at the Yantai University, Yantai, China. His research interests include blockchain, and machine learning with graphs.
Weishuai Che is currently pursuing the M.Sc. degree with the School of Computer and Control Engineering, Yantai University, Yantai, China. His current research interests include blockchain and machine learning with graphs.
Shenqiang Wang is currently pursuing the M.Sc. degree with the School of Computer and Control Engineering, Yantai University, Yantai, China. His current research interests include blockchain and machine learning with graphs.

Funding
This work was supported in part by the National Natural Science Foundation of China under Grant 62272405, School and Locality Integration Development Project of Yantai City(2022), the Youth Innovation Science and Technology Support Program of Shandong Provincial under Grant 2021KJ080, the Natural Science Foundation of Shandong Province under Grant ZR2022MF238, Yantai Science and Technology Innovation Development Plan Project under Grant 2022XDRH023.

Availability of data and materials
The data that support the findings of this study are available from the corresponding author Haoyu Yin, upon reasonable request.

Declarations

Ethics approval and consent to participate
This material is the authors' own original work, which has not been previously published elsewhere. The paper is not currently being considered for publication elsewhere.

Consent for publication
Not applicable.

Competing interests
The authors declare no competing interests.

Received: 20 March 2023 Accepted: 19 May 2023
Published online: 15 June 2023

References
1. Alassad M, Spann B, Agarwal N (2021) Combining advanced computational social science and graph theoretic techniques to reveal adversarial information operations. Inf Process Manag 58(1):102385
2. Hussain MJ, Wasti SH, Huang G, Wei L, Jiang Y, Tang Y (2020) An approach for measuring semantic similarity between wikipedia concepts using multiple inheritances. Inf Process Manag 57(3):102188

3. Gainza P, Sverrisson F, Monti F, Rodola E, Boscaini D, Bronstein M, Correia B (2020) Deciphering interaction fingerprints from protein molecular surfaces using geometric deep learning. Nat Methods 17(2):184–192
4. Zhou X, Li Y, Liang W (2021) Cnn-rnn based intelligent recommendation for online medical pre-diagnosis support. IEEE/ACM Trans Comput Biol Bioinforma 18(3):912–921. https://doi.org/10.1109/TCBB.2020.2994780
5. Gong W, Zhang X, Chen Y, He Q, Beheshti A, Xu X, Yan C, Qi L (2022) Dawar: Diversity-aware web apis recommendation for mashup creation based on correlation graph. In: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval. New York: ACM, pp 395–404
6. Nguyen GH, Lee JB, Rossi RA, Ahmed NK, Koh E, Kim S (2018) Continuous-time dynamic network embeddings. Companion Proceedings of the The Web Conference 2018:969–976
7. Heidari F, Papagelis M (2020) Evolving network representation learning based on random walks. Appl Netw Sci 5:1–38
8. Qi L, Chi X, Zhou X, Liu Q, Dai F, Xu X, Zhang X (2022) Privacy-aware data fusion and prediction for smart city services in edge computing environment. In: 2022 IEEE International Conferences on Internet of Things (iThings) and IEEE Green Computing & Communications (GreenCom) and IEEE Cyber, Physical & Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics), IEEE, pp 9–16
9. Mei JP, Lv H, Yang L, Li Y (2019) Clustering for heterogeneous information networks with extended star-structure. Data Min Knowl Disc 33:1059–1087
10. Pareja A, Domeniconi G, Chen J, Ma T, Suzumura T, Kanezashi H, Kaler T, Schardl T, Leiserson C (2020) Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In: Proceedings of the AAAI Conference on Artificial Intelligence. Menlo Park: AAAI Press, pp 5363–5370
11. Chen F, Wang YC, Wang B, Kuo CCJ (2020) Graph representation learning: a survey. APSIPA Trans Signal Inf Process 9:15
12. Trivedi R, Farajtabar M, Biswal P, Zha H (2019) Dyrep: Learning representations over dynamic graphs. In: International conference on learning representations. [Online].
13. Perozzi B, Al-Rfou R, Skiena S (2014) Deepwalk: Online learning of social representations. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. New York: ACM, pp 701–710
14. Grover A, Leskovec J (2016) node2vec: Scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. New York: ACM, pp 855–864
15. Zhou X, Liang W, Li W, Yan K, Shimizu S, Kevin I, Wang K (2021) Hierarchical adversarial attacks against graph-neural-network-based IoT network intrusion detection system. IEEE Internet Things J 9(12):9310–9319
16. Zhou X, Xu X, Liang W, Zeng Z, Yan Z (2021) Deep-learning-enhanced multitarget detection for end-edge-cloud surveillance in smart IoT. IEEE Internet Things J 8(16):12588–12596
17. Wang W, Wang Y, Duan P, Liu T, Tong X, Cai Z (2022) A triple real-time trajectory privacy protection mechanism based on edge computing and blockchain in mobile crowdsourcing. IEEE Trans Mob Comput, pp 1–18
18. Xu X, Zhang X, Gao H, Xue Y, Qi L, Dou W (2019) Become: Blockchain-enabled computation offloading for IoT in mobile edge computing. IEEE Trans Ind Inform 16(6):4187–4195
19. Xu X, Jiang Q, Zhang P, Cao X, Khosravi MR, Alex LT, Qi L, Dou W (2022) Game theory for distributed iov task offloading with fuzzy neural network in edge computing. IEEE Trans Fuzzy Syst 30(11):4593–4604
20. Zhou X, Liang W, Shimizu S, Ma J, Jin Q (2020) Siamese neural network based few-shot learning for anomaly detection in industrial cyber-physical systems. IEEE Trans Ind Inform 17(8):5790–5798
21. Liang W, Hu Y, Zhou X, Pan Y, Kevin I, Wang K (2021) Variational few-shot learning for microservice-oriented intrusion detection in distributed industrial IoT. IEEE Trans Ind Inform 18(8):5087–5095
22. Lu Z, Wang Y, Tong X, Mu C, Chen Y, Li Y (2021) Data-driven many-objective crowd worker selection for mobile crowdsourcing in industrial IoT. IEEE Trans Ind Inform 19(1):531–540
23. Makarov I, Kiselev D, Nikitinsky N, Subelj L (2021) Survey on graph embeddings and their applications to machine learning problems on graphs. PeerJ Comput Sci 7
24. Barros CD, Mendonça MR, Vieira AB, Ziviani A (2021) A survey on embedding dynamic graphs. ACM Comput Surv (CSUR) 55(1):1–37
25. Wang Y, Liu Z, Xu J, Yan W (2022) Heterogeneous network representation learning approach for ethereum identity identification. IEEE Trans Comput Soc Syst, pp 890–899
26. Liu Z, Yang D, Wang S, Su H (2022) Adaptive multi-channel bayesian graph attention network for iot transaction security. Digit Commun Netw, pp 1–20
27. Liu Z, Yang D, Wang Y, Lu M, Li R (2023) Egnn: Graph structure learning based on evolutionary computation helps more in graph neural networks. Appl Soft Comput 135:110040
28. Zhang H, Lu G, Zhan M, Zhang B (2022) Semi-supervised classification of graph convolutional networks with laplacian rank constraints. Neural Process Lett 54(4):2645–2656
29. You J, Ying R, Ren X, Hamilton W, Leskovec J (2018) Graphrnn: Generating realistic graphs with deep auto-regressive models. In: International conference on machine learning, PMLR, pp 5708–5717
30. Velickovic P, Cucurull G, Casanova A, Romero A, Lio P, Bengio Y (2017) Graph attention networks. Stat 1050:20
31. Jin G, Liu C, Chen X (2021) Adversarial network integrating dual attention and sparse representation for semi-supervised semantic segmentation. Inf Process Manag 58(5):102680
32. Yun S, Jeong M, Kim R, Kang J, Kim HJ (2019) Graph transformer networks. Adv Neural Inf Process Syst 32:1–11
33. Li R, Liu Z, Ma Y, Yang D, Sun S (2022) Internet financial fraud detection based on graph learning. IEEE Trans Comput Soc Syst, 1394–1401
34. Goyal P, Chhetri SR, Canedo A (2020) dyngraph2vec: Capturing network dynamics using dynamic graph representation learning. Knowl-Based Syst 187(104):816
35. Hisano R (2018) Semi-supervised graph embedding approach to dynamic link prediction. In: International Workshop on Complex Networks, Springer, pp 109–121
36. Wu Z, Zhan M, Zhang H, Luo Q, Tang K (2022) Mtgcn: A multi-task approach for node classification and link prediction in graph data. Inf Process Manag 59(3):102902
37. Haddad M, Bothorel C, Lenca P, Bedart D (2019) Temporalnode2vec: Temporal node embedding in temporal networks. In: International Conference on Complex Networks and Their Applications, Springer, pp 891–902
38. Hu L, Li C, Shi C, Yang C, Shao C (2020) Graph neural news recommendation with long-term and short-term interest modeling. Inf Process Manag 57(2):102142
39. Chen L, Tang X, Chen W, Qian Y, Li Y, Zhang Y (2021) Dacha: A dual graph convolution based temporal knowledge graph representation learning method using historical relation. ACM Trans Knowl Disc Data (TKDD) 16(3):1–18
40. Liu Z, Huang C, Yu Y, Dong J (2021) Motif-preserving dynamic attributed network embedding. Proc Web Conference 2021:1629–1638
41. Cui Z, Li Z, Wu S, Zhang X, Liu Q, Wang L, Ai M (2022) Dygcn: Efficient dynamic graph embedding with graph convolutional network. IEEE Trans Neural Netw Learn Syst, pp 1–12
42. Goel R, Kazemi SM, Brubaker M, Poupart P (2020) Diachronic embedding for temporal knowledge graph completion. In: Proceedings of the AAAI Conference on Artificial Intelligence. Menlo Park: AAAI Press, pp 3988–3995
43. Liu Y, Wu H, Rezaee K, Khosravi MR, Khalaf OI, Khan AA, Ramesh D, Qi L (2022) Interaction-enhanced and time-aware graph convolutional network for successive point-of-interest recommendation in traveling enterprises. IEEE Trans Ind Inf 19(1):635–643
44. Huang S, Bao Z, Li G, Zhou Y, Culpepper JS (2020) Temporal network representation learning via historical neighborhoods aggregation. In: 2020 IEEE 36th International Conference on Data Engineering (ICDE), IEEE, pp 1117–1128
45. Makarov I, Savchenko A, Korovko A, Sherstyuk L, Severin N, Kiselev D, Mikheev A, Babaev D (2022) Temporal network embedding framework with causal anonymous walks representations. PeerJ Comput Sci 8:858
46. Qu L, Zhu H, Duan Q, Shi Y (2020) Continuous-time link prediction via temporal dependent graph neural network. Proceedings of The Web Conference 2020:3026–3032
47. Chen H, Xiong Y, Zhu Y, Yu PS (2021) Highly liquid temporal interaction graph embeddings. Proceedings of the Web Conference 2021:1639–1648
48. Ma Y, Guo Z, Ren Z, Tang J, Yin D (2020) Streaming graph neural networks. In: Proceedings of the 43rd International ACM SIGIR Conference on

Research and Development in Information Retrieval. New York: ACM, pp 719–728
49. Zhang Z, Bu J, Ester M, Zhang J, Yao C, Li Z, Wang C (2020) Learning temporal interaction graph embedding via coupled memory networks. Proceedings of the web conference 2020:3049–3055
50. Liu W, Li H, Xie B (2018) Real-time graph partition and embedding of large network. 2018 18th IEEE/ACM International Symposium on Cluster. Cloud and Grid Computing (CCGRID), IEEE, pp 432–441
51. Goranci G, Räcke H, Saranurak T, Tan Z (2021) The expander hierarchy and its applications to dynamic graph algorithms. In: Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA), SIAM, pp 2212–2228
52. Wan Y, Yuan C, Zhan M, Chen L (2022) Robust graph learning with graph convolutional network. Inf Process Manag 59(3):102916
53. Mohan A, Pramod K (2022) Temporal network embedding using graph attention network. Complex Intell Syst 8(1):13–27
54. Zhou L, Yang Y, Ren X, Wu F, Zhuang Y (2018) Dynamic network embedding by modeling triadic closure process. In: Proceedings of the AAAI conference on artificial intelligence. Menlo Park: AAAI Press

Publisher's Note