Temporal Link Prediction: A Unified Framework, Taxonomy, and Review

MENG QIN and DIT-YAN YEUNG, Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong SAR

Dynamic graphs serve as a generic abstraction and description of the evolutionary behaviors of various complex systems (e.g., social networks and communication networks). Temporal link prediction (TLP) is a classic yet challenging inference task on dynamic graphs, which predicts possible future linkage based on historical topology. The predicted future topology can be used to support some advanced applications on real-world systems (e.g., resource pre-allocation) for better system performance. This survey provides a comprehensive review of existing TLP methods. Concretely, we first give the formal problem statements and preliminaries regarding data models, task settings, and learning paradigms that are commonly used in related research. A hierarchical fine-grained taxonomy is further introduced to categorize existing methods in terms of their data models, learning paradigms, and techniques. From a generic perspective, we propose a unified encoder-decoder framework to formulate all the methods reviewed, where different approaches only differ in terms of some components of the framework. Moreover, we envision serving the community with an open-source project OpenTLP\(^1\) that refactors or implements some representative TLP methods using the proposed unified framework and summarizes other public resources. As a conclusion, we finally discuss advanced topics in recent research and highlight possible future directions.

CCS Concepts: • General and reference → Surveys and overviews; • Networks → Network dynamics; • Information systems → Temporal data;

Additional Key Words and Phrases: Dynamic graphs, temporal link prediction, dynamic link prediction, complex network analysis

ACM Reference format:
Meng Qin and Dit-Yan Yeung. 2023. Temporal Link Prediction: A Unified Framework, Taxonomy, and Review. ACM Comput. Surv. 56, 4, Article 89 (November 2023), 40 pages. https://doi.org/10.1145/3625820

1 INTRODUCTION

For various complex systems (e.g., social networks, biology networks, and communication networks), graphs provide a generic abstraction to describe system entities and their relationships.

\(^1\)We will open source and constantly update OpenTLP at https://github.com/KuroginQin/OpenTLP.
For instance, one can abstract each entity as a node (vertex) and represent the relationship between a pair of entities as an edge (link) between the corresponding node pair. Each edge can also be associated with a weight to encode additional information about the interactions between system entities (e.g., trust rating between users \([1, 2]\) and traffic between telecommunication devices \([3, 4]\)).

Dynamic graphs, which can be represented as sequences of snapshots or time-induced edges, are widely used to describe behaviors of systems that change over time \([5]\). Temporal link prediction (TLP) is a classic yet challenging inference task on dynamic graphs. It aims at predicting possible linkage in specific future time steps based on the observed historical topology, playing an essential role in revealing the dynamic nature of systems and pre-allocating key resources (e.g., caches, CPU time, and communication channels) for better system performance \([6, 7]\).

The predicted future topology can also be used to support various advanced applications on real-world systems including (i) friend and next item recommendation in online social networks and media \([8, 9]\), (ii) intrusion detection in enterprise Internet \([10]\), (iii) channel allocation in wireless internet-of-things networks \([11]\), (iv) burst traffic detection and dynamic routing in optical networks \([12, 13]\), as well as (v) dynamics simulation and conformational analysis of molecules \([14]\).

As an extension of the conventional link prediction on static graphs \([15, 16]\), TLP is a more difficult task due to the following challenges. First, it is hard to capture the spatial-temporal characteristics of a dynamic graph, including the topology structures (e.g., interactions between nodes) in each time step and evolving patterns across successive time steps (e.g., changes of node interactions), which are usually complex and non-linear \([17]\). Second, the behaviors of some real-world systems may vary rapidly. Most of them also have the requirements of real-time inference. It is challenging to achieve fine-grained representations and predictions of the rapid variation while satisfying real-time constraints with low complexities. Third, most existing inference techniques on dynamic graphs have simple problem statements (e.g., TLP on unweighted graphs with fixed known node sets \([18, 19]\)). Some advanced settings from real-world systems (e.g., prediction of weighted links between previously unobserved nodes) are not fully studied in recent research.

To the best of our knowledge, there are a series of related surveys published in recent years with different focuses. Kazemi et al. \([20]\), Xue et al. \([21]\), and Barros et al. \([22]\) gave overviews of existing dynamic network embedding (DNE) techniques, which learn a low-dimensional vector representation (a.k.a. embedding) for each node with the evolving patterns of graph topology and other side information (e.g., node attributes) preserved. The derived embedding can then be used to support various downstream tasks including TLP. Also from the perspective of DNE, Skarding et al. \([23]\) reviewed recent techniques of graph neural networks (GNNs) for dynamic graphs. However, there remain gaps between the research on DNE and TLP. On the one hand, some classic TLP approaches \([24, 25]\) are not based on the DNE framework. On the other hand, some DNE methods \([26–28]\) are task-dependent with model architectures and objectives designed only for a specific setting of TLP. Moreover, most task-independent DNE techniques can only support simple TLP settings based on some common but naive strategies (e.g., treating the prediction of unweighted links as binary edge classification \([19, 29, 30]\)). The aforementioned surveys lack detailed discussions regarding whether and how a DNE method can be used to handle different settings of TLP. This survey covers both (i) classic TLP methods that do not rely on DNE and (ii) representative DNE approaches that can support TLP. In particular, for DNE-based techniques, we focus on how they can support TLP and why they cannot tackle some specific settings, highlighting their limitations.

Haghani et al. \([31]\) and Divakaran et al. \([32]\) reviewed representative TLP methods only based on the techniques they used but existing TLP techniques may differ in terms of multiple aspects.
(e.g., data models, learning paradigms, and task settings). We aim at giving a finer-grained description of existing TLP approaches covering multiple aspects via a unified framework. The major contributions of this survey can be summarized as follows.

- We propose a hierarchical taxonomy to categorize existing TLP methods in terms of the (i) data models, (ii) learning paradigms, and (iii) techniques used to handle the dynamic topology. Compared with existing surveys, the proposed taxonomy is a finer-grained description of existing approaches covering multiple aspects.

- From a generic perspective, we introduce a unified encoder-decoder framework to formulate all the TLP methods reviewed in this survey. In this framework, each method can be described by (i) an encoder, (ii) a decoder, and (iii) a loss function. It is expected that different methods only differ in terms of these three components.

- By using the proposed unified framework, we refactor and implement some representative TLP methods and serve the community with an open-source project OpenTLP (https://github.com/KuroginQin/OpenTLP). This project also summarizes some other public resources regarding TLP and will be constantly updated.

- In addition, some typical advanced research topics, future directions, quality evaluation criteria, applications, and datasets of TLP are also discussed in this survey.

In the rest of this survey, we give the problem statements and preliminaries regarding (i) data models of dynamic graphs, (ii) commonly-used task settings of TLP, and (iii) learning paradigms of recent research in Section 2. The unified encoder-decoder framework is introduced in the same section. In Section 3, we review representative TLP methods based on a fine-grained hierarchical taxonomy. Section 4 summarizes advanced topics in recent research and highlights possible future directions. Finally, Section 5 concludes this survey. We leave additional descriptions regarding the quality evaluation, detailed techniques, applications, and public datasets of TLP in supplementary materials.

2 PROBLEM STATEMENTS & PRELIMINARIES

In this survey, we consider the TLP on undirected homogeneous dynamic graphs, which covers the settings of most related research. Other inference tasks on directed heterogeneous graphs (e.g., knowledge graphs [33, 34]) are not included. For convenience, we summarize the major notations and abbreviations used in this survey in supplementary materials. In the rest of this section, we introduce the (i) data models of dynamic graphs as well as (ii) task settings, (iii) quality evaluation criteria, and (iv) learning paradigms that are commonly used in related research. From a generic perspective, a unified encoder-decoder framework is proposed to formulate all the TLP methods reviewed in this survey.

2.1 Data Models

Existing graph inference techniques usually adopt two data models to describe the dynamic topology, which are the (i) evenly-sampled snapshot sequence description (ESSD) and (ii) unevenly-sampled edge sequence description (UESD).

Definition 2.1 (ESSD). A dynamic graph can be represented as a sequence of snapshots $G = (G_1, G_2, \ldots, G_T)$ over a set of time steps $\{1, 2, \ldots, T\}$, where the time interval between successive snapshots is assumed to be regular. Each snapshot can be described as a tuple $G_t = (V_t, E_t)$, where $V_t = \{v_{t1}, v_{t2}, \ldots, v_{tN_t}\}$ is the set of nodes (with $v_{ti}$ denoting a node in $G_t$); $E_t = \{(v_{ti}, v_{tj}), w\)\mid v_{ti}, v_{tj} \in V_t, w \in \mathbb{R}^+\}$ is the set of edges with each edge $(v_{ti}, v_{tj})$ associated with a weight $w$. For unweighted graphs, we omit $w$ and use $E_t = \{(v_{ti}, v_{tj})\}$ to denote the edge...
Fig. 1. Examples of the (a) evenly-sampled snapshot sequence description (ESSD) and (b) unevenly-sampled edge sequence description (UESD) of a dynamic graph.

set. Some methods also assume that graph attributes are available, where each snapshot $G_t$ is associated with an attribute map $\mathcal{A}_t = \{\phi(v^t_1), \ldots, \phi(v^t_N_t)\}$ with $\phi(v^t_i)$ mapping each node $v^t_i$ to its attributes.

In general, we can use an adjacency matrix $A_t \in \mathbb{R}^{N_t \times N_t}$ to describe the topology of each snapshot $G_t$. For unweighted graphs, $(A_t)_{ij} = (A_t)_{ji} = 1$ when there is an edge between node pair $(v^t_i, v^t_j)$ and $(A_t)_{ij} = (A_t)_{ji} = 0$ otherwise. For weighted graphs, $(A_t)_{ij} = (A_t)_{ji} = w > 0$ denotes the weight of edge $(v^t_i, v^t_j)$ while $(A_t)_{ij} = (A_t)_{ji} = 0$ indicates that there is no edge between $(v^t_i, v^t_j)$. Graph attributes of snapshot $G_t$ can be described by an attribute matrix $X_t \in \mathbb{R}^{N_t \times M}$, where the $i$th row $(X_t)_i$ denotes the attributes of node $v^t_i$. In the rest of this survey, $\{A_t, X_t\}$ are used to describe the ESSD-based topology and attributes unless otherwise stated.

An example of ESSD for unweighted graphs is illustrated in Figure 1(a), where each snapshot $G_t$ describes the behavior of a system (e.g., data transmission in communication networks and user interactions in online social networks) at time step $t$ and successive snapshots have the same time interval (e.g., 1 second). When abstracting a real-world system via ESSD, one needs to manually select a fixed interval (or corresponding sampling rate) between successive time steps (i.e., snapshots). Accordingly, we can execute one prediction operation once it comes to a new time step. To fully describe the system behavior, the time interval is usually set to be the minimum duration of interactions in a system, which may result in high space complexities and many redundant descriptions of topology in applications with rapid variations. Therefore, ESSD is usually adopted as a coarse-grained description of dynamic graphs. Some other approaches use the UESD of dynamic graphs, which can describe system behaviors in a fine-grained manner.

**Definition 2.2 (UESD).** A dynamic graph can also be represented as a tuple $G_\Gamma = (V_\Gamma, E_\Gamma, \Gamma)$, where $\Gamma = \{t_1, t_2, \ldots\}$ is the set of time steps with $t_s \in \mathbb{R}^+$ as the time of the $s$th sampling and $t_1 < t_2 < \ldots < t_{|\Gamma|}$; $V_\Gamma = \{v_1, v_2, \ldots, v_{N_\Gamma}\}$ is the set of nodes observed during time period $\Gamma$; $E_\Gamma = \{((v_i, v_j), w, t_e) | v_i, v_j \in V_\Gamma, w \in \mathbb{R}^+, t_e \in \Gamma\}$ is the set of edges during $\Gamma$. In $E_\Gamma$, each edge $(v_i, v_j)$ is associated with a weight $w$ and a time step $t_e$, representing that there is an edge between node pair $(v_i, v_j)$ with weight $w$ at time step $t_e$. For unweighted graphs, we omit $w$ and use $E_\Gamma = \{((v_i, v_j), t_e) \}$ to denote the edge set. $t_e$ can be defined on the continuous domain. The interval between edges observed in two successive time steps can also be irregular. Some methods assume that
graph attributes $\mathcal{A}_\Gamma = \{\varphi(v_i, t) | t \in \Gamma, v_i \in V_\Gamma\}$ are available, where $\varphi(v_i, t)$ maps a node $v_i$ to its attributes observed at time step $t$.

An example of UESD for unweighted graphs is demonstrated in Figure 1(b), where each edge $(v_i, v_j)$ is associated with one or more positive numbers, implying that the interaction (e.g., data transmission and message communication) between system entities $v_i$ and $v_j$ is observed at corresponding time step. When using UESD, we sample a corresponding edge once there is a new interaction in the system without manually specifying the sampling rate. Hence, the interval between two successive time steps can be irregular, which makes it space-efficient to be a fine-grained description of dynamic graphs without the redundant descriptions of topology in ESSD. However, we still need to set a proper execution frequency of TLP for UESD.

Different from ESSD, UESD cannot use simple matrix representations of dynamic graphs (e.g., adjacency matrices). Hence, some mature matrix-based techniques (e.g., matrix factorization [35, 36]) cannot be applied to tackle the TLP with UESD. In contrast, most UESD-based methods rely on several continuous-time stochastic processes (e.g., Hawkes process [37, 38]) to capture the evolution of graph topology. However, these stochastic processes are still inapplicable to some advanced tasks that conventional matrix representations can easily address (e.g., inference on weighted dynamic graphs) due to the lack of related studies.

Some literature uses terminologies different from the aforementioned definitions. We argue that some of these terminologies cannot precisely describe the two data models. For instance, [19, 21, 39] defined the first and second data models as discrete(time) and continuous(-time) descriptions. Although the time index associated with each edge is defined on a continuous domain in the second data model, the edge sequence $E_\Gamma$ that describes the dynamic topology is still discrete. Therefore, the term “continuous(-time)” may be ambiguous in some cases. [40, 41] described the second data model using the term “streaming”. Nevertheless, “streaming” is usually used to describe a process with the continual arrival of events, while each edge in $E_\Gamma$ is not required to be continually generated. In fact, the major difference between the first and second data models is whether the interval between two successive time steps is irregular (i.e., unevenly sampled). In conclusion, we believe that ESSD and UESD can define the two data models more precisely.

### 2.2 Task Settings

Existing TLP methods may have different hypotheses and settings regarding the variation of node sets and availability of attributes in a dynamic graph. We categorize task settings of TLP into two levels with different degrees of difficulty based on their assumptions regarding the variation of node sets. In the rest of this survey, we use $\tau$ to represent the index of current time step. $L$ denotes the number of historical time steps or the historical time interval (a.k.a. window size) considered in a TLP method. $\Delta$ is the number of future time steps or the future time interval for prediction. For ESSD, $\mathcal{U}_d^\tau$ is adopted as a simplified notation of sequence $(\mathcal{U}_{s+1}^\tau, \mathcal{U}_{s+2}^\tau, \ldots, \mathcal{U}_d^\tau)$ w.r.t. a variable $\mathcal{U}$ (e.g., $\Gamma_{\tau-L}^\tau$ and $\mathcal{A}_{\tau-L}^{\tau+\Delta}$). For UESD, let $\Gamma(s, d) = \{t | s < t \leq d\}$ be the set of sampling time steps between $(s, d)$. $\mathcal{U}_\Gamma(s, d)$ denotes the sequence of a variable $\mathcal{U}$ associated with time steps in $\Gamma(s, d)$ (e.g., $\Gamma_{(r-L, r)}(r, r+\Delta)$).

**Definition 2.3 (TLP Level-1).** Level-1 assumes that the node set is known and fixed for all the time steps in a dynamic graph. Namely, there is no addition or deletion of nodes as the topology evolves. For ESSD, level-1 takes snapshots $\Gamma_{\tau-L}^\tau$ w.r.t. previous $L$ time steps and attributes $\mathcal{A}_{\tau-L}^{\tau+\Delta}$ (if available) as inputs and then predicts the topology w.r.t. next $\Delta$ time steps, which can be formulated as

$$\tilde{G}_{\tau+\Delta}^\tau = f_{\text{TLP}}(\Gamma_{\tau-L}^\tau, \mathcal{A}_{\tau-L}^{\tau+\Delta}),$$

(1)
with $\tilde{G}_{t+\Delta}^\tau$ as the prediction result. For UESD, given historical topology $G_{\Gamma(t-L,t)}$ and attributes $\mathcal{A}_{\Gamma(t-L,t+\Delta)}$ (if available), we aim at predicting the topology w.r.t. next $\Delta$ time steps. It can be formulated as

$$\tilde{G}_{\Gamma(t+\Delta)}^\tau = f_{\text{TLP}}(G_{\Gamma(t-L,t)}, \mathcal{A}_{\Gamma(t-L,t+\Delta)}),$$

where $\tilde{G}_{\Gamma(t+\Delta)}^\tau$ denotes the prediction result.

**Definition 2.4 (TLP Level-2).** Level-2 assumes that the node set can be non-fixed and can evolve over time, allowing the deletion and addition of nodes. In this setting, the prediction includes not only the future topology induced by old (i.e., previously observed) nodes but also edges (i) between an old node and a new (i.e., previously unobserved) node or (ii) between two new nodes. For ESSD, level-2 takes historical snapshots $G_{t-L}^\tau$, node sets $V_{t+\Delta}^\tau$ w.r.t. next $\Delta$ snapshots, and attributes $\mathcal{A}_{t-L}^\tau$ (if available) as inputs and then derives the prediction result $\tilde{G}_{t+\Delta}^\tau$ induced by $V_{t+\Delta}^\tau$ via

$$\tilde{G}_{t+\Delta}^\tau = f_{\text{TLP}}(G_{t-L}^\tau, V_{t+\Delta}^\tau, \mathcal{A}_{t-L}^\tau).$$

For UESD, given the historical topology $G_{\Gamma(t-L,t)}$, future node set $V_{\Gamma(t,t+\Delta)}$, and attributes $\mathcal{A}_{\Gamma(t-L,t+\Delta)}$ (if available), we can formulate the TLP in level-2 as

$$\tilde{G}_{\Gamma(t,t+\Delta)} = f_{\text{TLP}}(G_{\Gamma(t-L,t)}, V_{\Gamma(t,t+\Delta)}, \mathcal{A}_{\Gamma(t-L,t+\Delta)}),$$

where $\tilde{G}_{\Gamma(t,t+\Delta)}$ denotes the prediction result induced by $V_{\Gamma(t,t+\Delta)}$.

In general, level-2 is a more challenging setting, with level-1 as a special case of level-2. All the methods reviewed in this survey can deal with level-1 but only some of them can tackle level-2. Figure 2 gives an example of level-2 with $L = 3, \Delta = 1$, and $V_{t+\Delta}^\tau = \{v_1, v_2, v_5, v_6\}$ for ESSD and UESD. Some literature [29, 42, 43] also divides the inference tasks on dynamic graphs into the transductive and inductive settings according to the variation of node sets. For TLP, the transductive setting only considers the prediction of edges induced by old (i.e., previously observed) nodes (e.g., $(v_1, v_2)$) at time step $(\tau + 1)$ in Figure 2. In contrast, the inductive setting considers predicted edges (i) between an old node and a new node (e.g., $(v_1, v_6)$ at $(\tau + 1)$) or (ii) between two new nodes (e.g., $(v_5, v_6)$ at $(\tau + 1)$). Some studies [29, 42, 43] separately treated and evaluated the three sources of prediction results, while level-2 defined in this survey covers all the results.

For ESSD, settings with $\Delta = 1$ and $\Delta > 1$ are defined as the one-step and multi-step prediction. Most related methods focus on the one-step prediction. For a given current time step $\tau$, some approaches let $L = \tau$, where all historical snapshots are used for TLP and window size $L$ increases as the graph evolves. Other methods use a fixed setting of $L$ as $\tau$ increases, where only a fixed number of previous snapshots are utilized for TLP.

### 2.3 Quality Evaluation

Most existing TLP approaches focus on the prediction on unweighted graphs determining the existence of links between each pair of nodes in a future time step, which can be considered as a binary edge classification problem. Hence, some classic metrics of binary classification can be used to...
measure the prediction quality, including accuracy, F1-score, receiver operating characteristic (ROC) curve, and area under the ROC curve (AUC).

TLP on weighted graphs is a more challenging case seldom considered in recent research, which should not only determine the existence of future links but also predict associated link weights. Therefore, metrics of binary classification cannot be applied to measure the prediction quality. Root-mean-square error (RMSE) and mean absolute error (MAE) are widely-used metrics for the prediction of weighted graphs, which measure the reconstruction error between the prediction result and ground-truth. Qin et al. [44] argued that conventional RMSE and MAE metrics cannot measure the ability of a TLP method to derive high-quality prediction results for weighted graphs and proposed two new metrics of mean logarithmic scale difference (MLSD) and mismatch rate (MR) to narrow this research gap.

Due to space limit, we leave details of the aforementioned quality metrics in supplementary materials.

2.4 Learning Paradigms

Existing TLP techniques usually follow three learning paradigms, which can be summarized as (i) direct inference (DI), (ii) online training and inference (OTI), as well as (iii) offline training and online generalization (OTOG).

Definition 2.5 (Direct Inference, DI). Typical DI methods extract some manually designed or heuristic features from historical topology. The extracted features are directly used to infer the result of one prediction operation, in which there are no training procedures since DI methods do not have model parameters to be optimized. Once it comes to a new time step, one can repeat the direct inference procedure to derive a new prediction result.

Most DI methods are easy to implement and have the potential to satisfy the real-time constraints of some systems because there are no time-consuming model optimization procedures. However, since DI approaches are still based on simple heuristics and intuitions, they may fail to capture complex and non-linear characteristics of dynamic topology. OTI and OTOG are more advanced paradigms that can automatically extract informative latent characteristics from the raw dynamic graphs via model optimization. Figure 3 demonstrates examples of OTI and OTOG.

Definition 2.6 (Online Training & Inference, OTI). OTI methods are usually designed only for one prediction operation including a training phase and an inference phase. For a given current time step $\tau$, we first optimize the TLP model (i.e., the training phase) according to the inputs of historical topology ($G^{\tau}_{\tau-L}$ or $G_{r-L,r}$) and attributes ($A^{\tau}_{\tau-L}$ or $A_{r-L,r}$ if available). Only after that, the TLP model can derive the prediction result $\tilde{G}^{\tau+\Delta}_{\tau}$ or $\tilde{G}_{r+\Delta}$ (i.e., the inference phase). When it comes to a new time step $(\tau + 1)$, one should repeat the training and inference phases from scratch, in order to derive the new prediction result $G^{\tau+\Delta+1}_{\tau+1}$ or $G_{r+\Delta+1}$.
Definition 2.7 (Offline Training & Online Generalization, OTOG). In the OTOG paradigm, the sequence of snapshots or edges of a dynamic graph is divided into a training set $\Omega$ and a test set $\Omega'$, where snapshots or edges in $\Omega$ should occur before those in $\Omega'$. The TLP model is first trained on $\Omega$ in an offline way, with snapshots or edges in $\Omega$ as inputs and training ground-truth. After that, one can derive the prediction result of each new time step $\tau$ w.r.t. $\Omega'$ by directly generalizing the trained model to $\Omega'$ without additional optimization (i.e., with model parameters fixed).

In summary, OTI methods do not adopt the strategy that splits a dynamic graph into the training and test sets. Instead, they continually optimize the TLP model as time step $\tau$ increases in an online manner, which can capture the latest evolving patterns. However, OTI approaches usually suffer from efficiency issues due to the high complexities of their online training and thus cannot be deployed to systems with real-time constraints. In contrast, the OTOG paradigm includes offline training and online generalization. It is usually assumed that one has enough time to fully train a TLP model in an offline way. The runtime of generating a prediction result only depends on the online generalization. Since there is no additional optimization when generalizing the model to a test set, OTOG methods have the potential to satisfy the real-time constraints of systems. Nevertheless, they may also have the risk of failing to catch up with the latest variation of dynamic graphs, especially when there is a significant difference between the training and test sets.

2.5 Unified Encoder-Decoder Framework

From a generic perspective, we introduce a unified encoder-decoder framework to formulate existing TLP methods, which includes (i) an encoder Enc($\cdot$), (ii) a decoder Dec($\cdot$), and (iii) a loss function $\mathcal{L}(\cdot)$. It is expected that different TLP methods reviewed in this survey only differ in terms of these three components.

The encoder Enc($\cdot$) takes (i) historical topology and (ii) attributes (if available) as inputs and then derives an intermediate representation $R$ that captures the key properties of inputs. It can be formulated as

$$ R = \text{Enc}(G^{\tau-L}_{\tau-L}, \mathcal{A}^{\tau-L}_{\tau-L}) \quad \text{and} \quad R = \text{Enc}(G_{\Gamma(\tau-L, \tau)}, \mathcal{A}_{\Gamma(\tau-L, \tau)}), $$

for ESSD and UESD. The decoder Dec($\cdot$) further takes (i) intermediate representation $R$, (ii) node sets w.r.t. future time steps (if level-2 is considered), and (iii) attributes (if available) as inputs and then derives the final prediction result, which can be described as

$$ G^{\tau+\Delta}_{\tau} = \text{Dec}(R, V^{\tau+\Delta}_{\tau}, \mathcal{A}^{\tau+\Delta}_{\tau}) \quad \text{and} \quad G_{\Gamma(\tau, \tau+\Delta)} = \text{Dec}(R, V_{\Gamma(\tau, \tau+\Delta)}, \mathcal{A}_{\Gamma(\tau, \tau+\Delta)}), $$

for ESSD and UESD. Both the encoder and decoder may include a set of parameters that can be optimized (learned) via a loss function $\mathcal{L}(\cdot)$ regarding the historical topology and attributes (if available). In the rest of this survey, $\delta$ is used to denote the set of learnable model parameters. The loss function and its optimization objective can be formulated as

$$ \min_{\delta} \mathcal{L}(G^{\tau-L}_{\tau-L}, \mathcal{A}^{\tau-L}_{\tau-L}; \delta) \quad \text{and} \quad \min_{\delta} \mathcal{L}(G_{\Gamma(\tau-L, \tau)}, \mathcal{A}_{\Gamma(\tau-L, \tau)}; \delta), $$

for ESSD and UESD. In the rest of this survey, we omit $\mathcal{A}^{\tau+\Delta}_{\tau-L}$ and $\mathcal{A}_{\Gamma(\tau-L, \tau+\Delta)}$ if attributes are not available. Furthermore, we omit $V^{\tau+\Delta}_{\tau}$ or $V_{\Gamma(\tau, \tau+\Delta)}$ if the node set is assumed to be fixed (i.e., TLP in level-1).

In some cases, the intermediate representation $R$ given by Enc($\cdot$) can be dynamic network embedding (DNE) (a.k.a. dynamic graph representation learning) [20–22] but is not limited to it.

Definition 2.8 (Dynamic Network Embedding, DNE). Let $V$ denote the set of all possible nodes. Given the historical topology (described by $G^{\tau-L}_{\tau-L}$ or $G_{\Gamma(\tau-L, \tau)}$) and attributes (described by $\mathcal{A}^{\tau-L}_{\tau-L}$
Fig. 4. Illustrations of (a) task-dependent and (b) task-independent DNE for TLP in terms of the unified encoder-decoder framework.

| Strategies  | Definitions                                                                 | Strategies  | Definitions                                                                 | Strategies  | Definitions                                                                 |
|-------------|------------------------------------------------------------------------------|-------------|------------------------------------------------------------------------------|-------------|------------------------------------------------------------------------------|
| Concatenation | $e_{ij} = [z_i | z_j]$                                                        | Hadamard Product | $e_{ij} = z_i \odot z_j$                                                      | Weighted-$l_2$ Norm | $e_{ij} = |z_i - z_j|^2$                                                   |
| Average     | $e_{ij} = (z_i + z_j)/2$                                                      | Weighted-$l_1$ Norm | $e_{ij} = |z_i - z_j|$                                                        |                                                                                   |

or $A_{\Gamma(\tau-L, \tau)}$ if available), DNE aims at learning a function $f_{\text{DNE}}: V \mapsto \mathbb{R}^d$ that maps each node $v_i \in V$ to a low-dimensional vector representation $z_i \in \mathbb{R}^d$ (a.k.a. node embedding) with $d \ll |V|$ as the embedding dimensionality. The derived embedding $\{z_i\}$ is expected to preserve the evolving patterns of dynamic topology and attributes (if available). For example, nodes $(v_i, v_j)$ with an edge at a time step close to $\tau$ are more likely to have similar representations $z_i$ and $z_j$ with close distance or high similarity. The derived $\{z_i\}$ can be used to support various downstream tasks on future topology including TLP.

Although there is a close relationship between DNE and TLP, there remain gaps between the research on these two tasks. First, some classic TLP methods (e.g., neighbor similarity [45] and graph summarization [24, 25] described in Section 3.2) are not based on DNE. This survey covers DNE techniques that can support TLP but is not limited to them.

Furthermore, as shown in Figure 4(a), some TLP approaches follow a task-dependent DNE framework with specific encoders and decoders designed only for TLP. In most cases, supervised losses regarding TLP are applied to optimize the DNE models (e.g., by minimizing the error between the topology reconstructed by $R$ and ground-truth) in an end-to-end manner. Therefore, these DNE-based approaches are optimized only for TLP.

Other DNE techniques are task-independent as illustrated in Figure 4(b), which optimize the model via unsupervised losses to derive embedding regardless of downstream tasks. For TLP, these methods rely on some commonly-used but naive designs of decoders to derive prediction results without specifying their own decoders. In these designs of decoders, an auxiliary edge embedding $e_{ij}$ is first derived for each pair of nodes $(v_i, v_j)$ using corresponding node embedding $(z_i, z_j)$. A downstream binary classifier (e.g., logistic regression) is then trained with $\{e_{ij}\}$ as inputs and finally outputs the probability that an edge $(v_i, v_j)$ appears in a future time step. Table 1 summarizes some commonly-used strategies to derive $\{e_{ij}\}$. However, these strategies may still fail to handle some advanced applications of TLP (e.g., prediction of weighted links) due to the binary output of the downstream classifier.

In summary, existing DNE-based methods may have different designs of encoders, decoders, and loss functions for TLP. Not all the designs can tackle some advanced settings (e.g., prediction of weighted links and TLP in level-2). In addition to the description of model architectures, this survey also highlights their limitations to TLP w.r.t. each component in the unified encoder-decoder framework, which is not covered in existing survey articles regarding DNE.
3 REVIEW OF TEMPORAL LINK PREDICTION METHODS

3.1 Overview of the Hierarchical Fine-Grained Taxonomy

As illustrated in Figure 5, we propose a hierarchical taxonomy that can describe a TLP method in a finer-grained manner covering multiple aspects, compared with those in existing surveys. The proposed taxonomy first categorizes existing methods according to their data models (i.e., ESSD and UESD) as described in Section 2.1. Both the data models can be further categorized based on learning paradigms (i.e., DI, OTI, and OTOG) defined in Section 2.4. Finally, each method is characterized based on the techniques used to handle the dynamic topology.

Table 2 summarizes details of all the methods reviewed in this survey. In addition to the data models and learning paradigms covered in our hierarchical taxonomy, we also highlight properties of the (i) ability to handle the variation of node sets (i.e., level-1 or -2), (ii) availability of node attributes, (iii) ability to capture and predict weighted links, (iv) setting of window size $L$, and (v) number of future time steps or time interval $\Delta$ for prediction. Since the abilities of some task-dependent DNE based methods to predict weighted links rely on the designs of their decoders and loss functions, we use “D/L-Dep” to denote that the original version of a method cannot support the weighted TLP but can be easily extended to tackle this setting by replacing the decoder or loss function.

The complexity of one prediction operation denoted as $O_P$ is also summarized in Table 2, where $O_M$ and $O_I$ denote the complexities of model optimization and inference; $n$ and $m$ are the numbers of nodes and attributes; $e$ is the maximum number of edges in training snapshots $G^\tau_{\tau-L}$ for ESSD; $d$ is the embedding dimensionality that is usually much smaller than $n$, $m$, and $e$; $I$ is the number of iterations in model optimization. As we consider the inference that estimates future links between all the possible $n^2$ node pairs, some methods are with $O_I \geq O(n^2)$. For DL-based methods, we could only use $O_{OPT}$ and $O_{FFP}$ to respectively represent their complexities of model optimization and inference (i.e., feedforward propagation of DL modules), because the complexities of some DL models are usually hard to analyze, which rely heavily on layer configurations, initialization, and optimization settings of optimizers, learning rates, and numbers of epochs. In general, $O_{FFP}$ of a method is larger than $O(n^2 dL)$. $O_{OPT}$ is usually assumed to be time-consuming. We can further speed up the overall runtime of these DL-based methods using parallel implementations and GPUs. Consistent with our discussions in Section 2.4, $O_P$ only includes $O_I$ (without
time-consuming model optimization) for DI and OTOG approaches, thus having the potential to satisfy the real-time constraints of systems.

As the properties of a TLP method largely depend on its data model and learning paradigm, we also summarize the advantages and disadvantages of different types of approaches according to the two aspects in Table 3, which are detailed later at the end of each subsection (see Sections 3.2.3, 3.3.2, 3.4.4, 3.5.3, and 3.6.3).

### 3.2 ESSD-Based DI Methods

Some ESSD-based TLP methods adopt the following hypothesis about the evolving patterns of dynamic graphs.

**Hypothesis 3.1.** Given a dynamic graph \( G = (G_1, G_2, \ldots, G_t, \ldots) \), snapshots close to the current time step \( \tau \) should have more contributions than those far away from \( \tau \) in the optimization and inference w.r.t. the prediction of \( G_{t+\Delta} \).

Typical techniques used in existing DI methods include neighbor similarity and graph summarization.

#### 3.2.1 Neighbor Similarity

Some classic TLP methods are based on neighbor similarities [45] of dynamic graphs. These approaches use several similarity measures between nodes in current snapshot \( G_\tau \) to predict next snapshot \( G_{\tau+1} \), with the window size set to \( L = 1 \). Consistent with **Hypothesis 3.1**, \( G_\tau \) is assumed to have properties closest to the snapshot \( G_{\tau+1} \) to be predicted. In our unified framework, the encoder and decoder of neighbor similarity-based methods can be described as

\[
\text{Enc}(G^\tau_{\tau-L}) \equiv G_\tau \quad \text{and} \quad (\tilde{A}_{\tau+1})_{ij} = \text{Dec}(v^\tau_i, v^\tau_j) \equiv \text{sim}(v^\tau_i, v^\tau_j),
\]

\[ \tag{8} \]

| Methods               | Data Models | Learning Paradigms | Level | Attributes | Link Weights | \( L \) | \( \Lambda \) | \( O_P \) | \( O_M \) | \( O_I \) |
|----------------------|-------------|--------------------|-------|------------|--------------|-------|----------|---------|----------|---------|
| Neighbor Similarity  | ESSD        | DI                 | 1     | N/A        | Unable       | 1     | 1        | \( O_I \) | 0        | \( O(n^2) \) to \( O(n^3) \) |
| Graph Summarization  | ESSD        | DI                 | 1     | N/A        | Able         | 1     | 1        | \( O_I \) | 0        | \( O(\epsilon L) \) |
| CRJMF [46]           | ESSD        | OTI                | 1     | Static     | Able         | 1     | 1        | \( O_M + O_I \) | \( O(\epsilon + \eta + \mu) \) | \( O(n^d) \) |
| TLSI [47]            | ESSD        | OTI                | 1     | N/A        | Able         | 1     | 1        | \( O_M + O_I \) | \( O(\epsilon + nd) \) | \( O(n^d) \) |
| MLjFE [48]           | ESSD        | OTI                | 1     | N/A        | Able         | 1     | 1        | \( O_M + O_I \) | \( O(\epsilon + \eta + \mu) \) | \( O(n^d) \) |
| GrNNFM [49]          | ESSD        | OTI                | 1     | N/A        | Able         | 1     | 1        | \( O_M + O_I \) | \( O(\epsilon + \eta + \mu) \) | \( O(n^d) \) |
| DeepEye [50]         | ESSD        | OTI                | 1     | N/A        | Able         | 1     | 1        | \( O_M + O_I \) | \( O(\epsilon + nd) \) | \( O(n^d) \) |
| TMF [51]             | ESSD        | OTI                | 1     | N/A        | Able         | 1     | 1        | \( O_M + O_I \) | \( O(\epsilon + nd) \) | \( O(n^d) \) |
| LIST [52]            | ESSD        | OTI                | 1     | N/A        | Unable       | 1     | 1        | \( O_I \) | \( O_{OPT} \) | \( O_{BP} \) |
| cTRBM [18]           | ESSD        | OTOG               | 1     | N/A        | Fixed        | 1     | 1        | \( O_I \) | \( O_{OPT} \) | \( O_{BP} \) |
| dynagraph2vec [27]   | ESSD        | OTOG               | 1     | N/A        | Fixed        | 1     | 1        | \( O_I \) | \( O_{OPT} \) | \( O_{BP} \) |
| DDNE [26]            | ESSD        | OTOG               | 1     | N/A        | D/L-Dep      | Fixed | 1     | \( O_I \) | \( O_{OPT} \) | \( O_{BP} \) |
| EvolveGCN [53]       | ESSD        | OTOG               | 2     | Dynamic    | D/L-Dep      | Fixed | 1     | \( O_I \) | \( O_{OPT} \) | \( O_{BP} \) |
| GCN-GAN [7]          | ESSD        | OTOG               | 1     | N/A        | Fixed        | 1     | 1        | \( O_I \) | \( O_{OPT} \) | \( O_{BP} \) |
| IDEA [44]            | ESSD        | OTOG               | 2     | Static     | Able         | 1     | 1        | \( O_I \) | \( O_{OPT} \) | \( O_{BP} \) |
| STGSN [28]           | ESSD        | OTOG               | 2     | Dynamic    | D/L-Dep      | Fixed | 1     | \( O_I \) | \( O_{OPT} \) | \( O_{BP} \) |
| DysSAT [30]          | ESSD        | OTOG               | 2     | Dynamic    | Fixed        | 1     | 1        | \( O_I \) | \( O_{OPT} \) | \( O_{BP} \) |
| CTDNE [19]           | UESSD       | OTI                | 1     | N/A        | Unable       | 1     | \( >0 \) | \( O_M + O_I \) | \( O_{OPT} \) | \( O(n^d) \) |
| HTNE [54]            | UESSD       | OTI                | 1     | N/A        | Unable       | 1     | \( >0 \) | \( O_M + O_I \) | \( O_{OPT} \) | \( O(n^d) \) |
| M2DNE [55]           | UESSD       | OTI                | 1     | N/A        | Unable       | 1     | \( >0 \) | \( O_M + O_I \) | \( O_{OPT} \) | \( O(n^d) \) |
| TGAT [29]            | UESSD       | OTOG               | 2     | Dynamic    | Unable       | \( >0 \) | \( O_I \) | \( O_{OPT} \) | \( O_{BP} \) | \( O_{BP} \) |
| CAY [43]             | UESSD       | OTOG               | 2     | Dynamic    | Unble        | \( >0 \) | \( O_I \) | \( O_{OPT} \) | \( O_{BP} \) | \( O_{BP} \) |
| DyRep [42]           | UESSD       | OTOG               | 2     | Dynamic    | Unble        | \( >0 \) | \( O_I \) | \( O_{OPT} \) | \( O_{BP} \) | \( O_{BP} \) |
| TREND [56]           | UESSD       | OTOG               | 2     | Static     | Unble        | \( >0 \) | \( O_I \) | \( O_{OPT} \) | \( O_{BP} \) | \( O_{BP} \) |
| GSNOP [57]           | UESSD       | OTOG               | 2     | Dynamic    | Unble        | \( >0 \) | \( O_I \) | \( O_{OPT} \) | \( O_{BP} \) | \( O_{BP} \) |
Inefficient Common Neighbor for

Unable,

\|OTOG \in \text{to support TLP on similarity measures definitions (1)}

Adamic-Adar (1) Easy to implement w/o loss functions for model optimization.

Unable to support TLP in level-2.

OTOG \in \text{to capture the latest evolving patterns of topology. (1)}

(2) Able to support TLP on weighted graphs.

EESD OTI

(1) Unable to capture non-linear characteristics of topology.

(2) Inefficient for applications w/ real-time constraints.

(3) Only supporting coarse-grained representations of dynamic topology.

(4) Unable to support TLP in level-2.

Table 4. Some Commonly-used Similarity Measures $\text{sim}(v^*_i, v^*_j)$ for neighbor similarity based Methods

Table 3. Summary of the Advantages and Disadvantages of Different Types of Methods

Data Models Learning Paradigms

| ESSD | DI | +Pros. (1) Easy to implement w/o loss functions for model optimization. (2) Having the potential to satisfy real-time constraints of prediction. -Cons. (1) Unable to capture complex non-linear characteristics of topology. (2) Only supporting coarse-grained representations of dynamic topology. (3) Unable to support TLP in level-2. |
|------|----|--------------------------------------------------|

| ESSD | OTI | +Pros. (1) Able to capture the latest evolving patterns of topology. (2) Able to support TLP on weighted graphs. -Cons. (1) Unable to capture non-linear characteristics of topology. (2) Inefficient for applications w/ real-time constraints. (3) Only supporting coarse-grained representations of dynamic topology. (4) Unable to support TLP in level-2. |
|------|-----|--------------------------------------------------|

| ESSD | OTOG | +Pros. (1) Having the potential to satisfy real-time constraints of prediction. (2) Most of the methods or their modified versions can support TLP on weighted graphs. (3) Some methods can even derive high-quality weighted prediction results. -Cons. (1) Only supporting coarse-grained representations of dynamic graphs. (2) Having the risk of failing to capture the latest evolving patterns. |
|------|------|--------------------------------------------------|

| UESD | OTI | +Pros. (1) Able to capture the latest evolving patterns of topology. (2) Able to support fine-grained representations of dynamic topology. -Cons. (1) Inefficient for applications w/ real-time constraints. (2) Unable to support TLP in level-2. (3) Unable to support TLP on weighted graphs. |
|------|-----|--------------------------------------------------|

| UESD | OTOG | +Pros. (1) Able to support fine-grained representations of dynamic topology. (2) Having the potential to satisfy real-time constraints of prediction. (3) Able to support TLP in level-2. -Cons. (1) Unable to support TLP on weighted graphs. (2) Having the risk of failing to capture the latest evolving patterns. |
|------|------|--------------------------------------------------|

where $\text{sim}(v^*_i, v^*_j)$ denotes a neighbor similarity measure between nodes $(v^*_i, v^*_j)$, which is also the output of decoder. The derived prediction result $(\hat{A}_{t+1})_{ij} = \text{sim}(v^*_i, v^*_j)$ is directly proportional to the probability that there is an edge between $(v^*_i, v^*_j+1)$ in the next snapshot.

Let $\text{Nei}(v^*_i)$ and $\text{SPth}(v^*_i, v^*_j)$ be the set of neighbors of $v^*_i$ and the shortest path between $(v^*_i, v^*_j)$. Some commonly-used neighbor similarity measures are summarized in Table 4. These measures are based on several intuitions that nodes $v^*_i$ and $v^*_j$ are more likely to form a link if (i) their neighbors $\text{Nei}(v^*_i)$ and $\text{Nei}(v^*_j)$ have a large overlap (with several normalization strategies) or (ii) the length of shortest path $|\text{SPth}(v^*_i, v^*_j)|$ is small. Let $\text{Pth}(v^*_i, v^*_j)$ be the set of paths between $(v^*_i, v^*_j)$ with length $k$. Katz index [58] in Table 4 is equivalent to $\sum_{k=1}^{\infty} \theta^k |\text{Pth}(v^*_i, v^*_j)|$, which is the discounted sum of the number of paths between $(v^*_i, v^*_j)$ with length $k \in [1, \infty)$. $\theta \in (0, 1)$ is a pre-set decaying factor.

Since there is no model optimization procedure for neighbor similarity based methods due to the DI paradigm, we do not need to define their loss functions. However, they derive prediction results only based on current snapshot $G_t$ (with $L = 1$), failing to explore the dynamic topology across snapshots. Furthermore, their similarity measures are usually designed only for unweighted topology, indicating that they cannot tackle the prediction of weighted links.
3.2.2 Graph Summarization. Graph summarization [24, 25] is another classic technique for inference tasks on dynamic graphs. It collapses historical snapshots $G^\tau_{t-L}$ into an auxiliary weighted snapshot $G_C$ via linear combination, which is expected to preserve key properties of dynamic topology. In our encoder-decoder framework, graph summarization can be described as

$$\text{Enc}(G^\tau_{t-L}) \equiv W_\tau \text{ and } \hat{A}_{t+1} = \text{Dec}(W_\tau) \equiv W_\tau,$$

where $W_\tau$ is the adjacency matrix of the collapsed snapshot $G_C$ and is directly adopted as the prediction result $\hat{A}_{t+1}$. For the prediction of unweighted graphs, each element $(\hat{A}_{t+1})_{ij}$ is directly proportional to the probability that there is an edge between $(v^\tau_{i+1}, v^\tau_{j+1})$. The normalized value of $(W_\tau)_{ij}$ (e.g., averaging $W_\tau$ over the window size $L$) can also be the predicted edge weight of $(v^\tau_{i+1}, v^\tau_{j+1})$ for weighted graphs.

Different variants of graph summarization only differ in terms of their encoders. Typical variants include the exponential kernel $K_E(\cdot)$ [24], inverse linear kernel $K_{IL}(\cdot)$ [25], and linear kernel $K_L(\cdot)$ [25], which are defined as

$$\text{Enc}(G^\tau_{t-L}) = K_E(A^\tau_{t-L}) \equiv \sum_{\tau=t-L+1}^\tau (1 - \theta)^{\tau-\theta} A_t,$$

$$\text{Enc}(G^\tau_{t-L}) = K_{IL}(A^\tau_{t-L}) \equiv \sum_{\tau=t-L+1}^\tau \frac{1}{(\tau - t + 1)} A_t,$$

$$\text{Enc}(G^\tau_{t-L}) = K_L(A^\tau_{t-L}) \equiv \sum_{\tau=t-L+1}^\tau \frac{L}{(\tau - t + 1)} A_t,$$

where $\theta \in [0, 1]$ is a tunable parameter. The three kernels adopt different decaying rates for adjacency matrices $A^\tau_{t-L}$ w.r.t. historical topology to ensure Hypothesis 3.1. Due to the DI paradigm, graph summarization does not have the loss function for model optimization. In some cases, one can combine neighbor similarity with graph summarization by applying neighbor similarities (see Table 4) to the predicted adjacency matrix $\hat{A}_{t+1}$ to refine the prediction result.

3.2.3 Summary of ESSD-Based DI Methods. For DI methods, there are no loss functions defined for model optimization in our encoder-decoder framework. Therefore, they are easy to implement and have the potential to satisfy the real-time constraints of applications. However, these methods cannot fully capture the complex non-linear characteristics of dynamic graphs because they still rely on simple intuitions and linear models. Furthermore, they can only support coarse-grained representations of dynamic graphs due to the limitations of ESSD. As the aforementioned approaches are based on heuristics designed for graphs with fixed node sets, they can only support the TLP in level-1, failing to tackle the variation of node sets.

3.3 ESSD-Based OTI Methods

3.3.1 Matrix Factorization. Most ESSD-based OTI methods combine matrix factorization techniques [35, 36] with Hypothesis 3.1. They decompose adjacency matrices $A^\tau_{t-L}$ or their transformations into low-dimensional matrices (e.g., $\arg \min_{U_t, V_t} \|A_t - U_t V_t^T\|_F^2$) with key properties of historical topology preserved. The derived latent matrices can be used to “reconstruct” the adjacency matrix of a future snapshot via an inverse process of matrix factorization (e.g., $\hat{A}_{t+1} = U_t V_t^T$).

Non-negative matrix factorization (NMF) [35, 59] is a typical technique adopted by related methods, where there are non-negative constraints on the latent matrices to be learned (e.g., $\arg \min_{U_t, V_t \geq 0} \|A_t - U_t V_t^T\|_F^2$). NMF-based approaches are usually optimized via specific multiplicative procedures [60] instead of classic additive optimization algorithms (e.g., gradient descent). We leave details regarding the model optimization of NMF in supplementary materials. In
our encoder-decoder framework, most NMF-based methods have similar definitions of decoders but differ from their encoders and loss functions.

(1) CRJMF. Gao et al. [46] proposed CRJMF, which extends graph summarization to incorporate additional node attributes and neighbor similarity using NMF. It assumes that node attributes, described by a matrix $X \in \mathbb{R}^{N \times M}$, are available and fixed for all snapshots. A similarity matrix $S \in \mathbb{R}^{N \times N}$ is also introduced to describe the neighbor similarity, where $S_{ij}$ is the common neighbor measure between $(v_i^t, v_j^t)$ as defined in Table 4. Given historical snapshots $G_{t-L}^t$ and fixed node attributes $A$, the encoder and loss function of CRJMF can be described as

$$\text{Enc}(G_{t-L}^t, A) \equiv \arg \min_{U \geq 0, V \geq 0, Y \geq 0} \mathcal{L}(G_{t-L}^t, A; U, V, Y) \equiv \left\| W_r - UYU^T \right\|_F^2 + \alpha \left\| X - UV^T \right\|_F^2 + \beta \text{tr}(U^T L S U),$$

(13)

where $(U \in \mathbb{R}^{N \times d}, V \in \mathbb{R}^{N \times d}, Y \in \mathbb{R}^{d \times d})$ are latent matrices to be optimized and outputs of the encoder; $W_r$ is derived from the exponential kernel of graph summarization defined in (10); $L_S = D_S - S$ is the Laplacian matrix of $S$; $D_S = \text{diag}(d_1^t, d_2^t, \ldots, d_N^t)$ is the degree diagonal matrix of $S$ with $d_i^t = \sum_j S_{ij}$; $\alpha$ and $\beta$ are parameters to adjust the second and third terms. In particular, the third term is defined as graph regularization [61], which can be rewritten as $\text{tr}(U^T L S U) = 0.5 \cdot \sum_{i,j=1}^N S_{ij} |U_{i.} - U_{j.}|_2^2$. It can apply additional regularization encoded in $S$ (i.e., the second-order neighbor similarity) to $U$, where larger $S_{ij}$ forces $(U_{i.}, U_{j.})$ to be closer in the latent space.

Since $U$ is shared by all the terms in (13), it is expected to jointly encode the properties of historical topology, node attributes, and neighbor similarity after model optimization, while variables $\{V, Y\}$ are auxiliary variables. Let $\{U^t, V^t, Y^t\}$ be the solution to objective (13). The decoder executes an inverse process of the first term in (13), which is defined as

$$\tilde{A}_{t+1} = \text{Dec}(U^t, Y^t) \equiv U^t Y^t U^{tT}.$$  

(14)

However, CRJMF still relies on graph summarization to explore dynamic topology, which simply collapses historical snapshots into an auxiliary weighted snapshot described by $W_r$. In contrast, some other NMF-based approaches directly extract latent characteristics from the raw dynamic topology.

(2) TLSI. Zhu et al. [47] proposed TLSI, which automatically learns latent representations regarding dynamic topology based on the temporal smoothness intuition that nodes change their representations smoothly over time. Given historical snapshots $G_{t-L}^t$, the encoder and loss function of TLSI are defined as

$$\text{Enc}(G_{t-L}^t) \equiv \arg \min_{U_{t-L} \geq 0} \mathcal{L}(G_{t-L}^t; U_{t-L}^t)$$

$$\equiv \sum_{t=t-L+1}^t \left\| A_t - U_t U_t^T \right\|_F^2 + \beta \sum_{t=t-L+2}^t \text{tr}(U_t U_{t-1}^T - I) \text{ s.t. } \text{tr}(U_t^T U_t - I) = 0,$$

(15)

where $(U_t \in \mathbb{R}^{N \times d})$ are latent matrices to be learned and outputs of the encoder, with $(U_t)_{i.}$ as the representation of node $v_i^t$. The first term is the standard symmetric NMF, which enables $U_t$ to encode the structural properties of snapshot $G_t$. Namely, nodes $(v_i^t, v_j^t)$ close to each other in $G_t$ should have close representations $((U_t)_{i.}, (U_t)_{j.})$. $\text{tr}(U_t^T U_{t-1} - I)$ is the temporal smoothness term of time step $t$ that penalizes each node for suddenly changing its representation, following the temporal smoothness intuition. In this setting, $(U_t)$ can also capture temporal characteristics, where successive snapshots have close latent representations. $\beta$ is a parameter to balance the objectives of NMF and temporal smoothness. The constraint $\text{tr}(U_t^T U_t - I) = 0$ ensures that $U_{t-L}^t$ are normalized for each node.
Let \( \{U_t^r\} \) be the solution to the aforementioned objective. The decoder derives the prediction result \( \hat{A}_{r+1} \) by executing an inverse process of symmetric NMF, which can be described as

\[
\hat{A}_{r+1} = \text{Dec}(U_r^r) \equiv U_r^r U_r^{rT}.
\]

(3) **MLjFE.** Also based on **temporal smoothness**, Ma et al. [48] proposed MLjFE. In addition to dynamic topology, a point-wise mutual information matrix \( M_t = \ln(\sum_i h(A_t)^i / h) \) is introduced to encode the high-order proximity of each snapshot \( G_t \) (i.e., multi-step neighbor similarities beyond the observable topology described by \( A_t \)), with \( h \) as the order to be specified. Given historical snapshots \( G_{r-L}^r \), the encoder and loss function can be described as

\[
\text{Enc}(G_{r-L}^r) \equiv \arg\min_{U_{r-L} \geq 0, V_{r-L} \geq 0} \mathcal{L}(G_{r-L}^r; U_{r-L}^r, V_{r-L}^r) \\
\equiv \sum_{t=r-L+1} \left[ ||A_t - U_t V_t^T||_F^2 + \alpha ||M_t - V_t Y_t^T||_F^2 \right] + \beta \sum_{t=r-L+2} ||U_t - U_{t-1}||_F \text{ s.t. } \text{tr}(V_t^T V_t - I) = 0
\]

where \( \{U_t \in \mathbb{R}^{N \times d}, V_t \in \mathbb{R}^{N \times d}, Y_t \in \mathbb{R}^{N \times d} \} \) are latent matrices to be learned; \( ||U_t - U_{t-1}||_F^2 \) is the **temporal smoothness term** with the same physical meaning as that in (15); \( \{\alpha, \beta\} \) are parameters to adjust the contributions of high-order proximity and temporal smoothness. Similar to (15), the constraint \( \text{tr}(V_t^T V_t - I) = 0 \) normalizes \( \{V_t\} \) for each node.

After model optimization, \( V_{r-L}^r \) shared by the first and second terms can comprehensively encode the temporal characteristics and high-order proximity of successive snapshots, while \( U_{r-L}^r \) and \( Y_{r-L}^r \) are auxiliary variables. Let \( \{U_t^r, V_t^r, Y_t^r\} \) be the solution to objective (17). The decoder of MLjFE derives the prediction result \( \hat{A}_{r+1} \) based on \( \{U_r^r, V_r^r\} \):

\[
\hat{A}_{r+1} = \text{Dec}(U_r^r, V_r^r) \equiv \sum_{t=r-L+1} \theta^{r-t} U_t^r V_t^{rT},
\]

where \( \theta \in [0, 1] \) is a pre-set time decaying factor to ensure **Hypothesis 3.1**.

(4) **GrNMF.** In addition to **temporal smoothness**, GrNMF [49] uses the NMF-based **graph regularization** [61] to explore evolving patterns of dynamic graphs. Given historical snapshots \( G_{r-L}^r \), the encoder and loss function of GrNMF are

\[
\text{Enc}(G_{r-L}^r) \equiv \arg\min_{U \geq 0, V \geq 0} \mathcal{L}(G_{r-L}^r; U, V) \equiv \left[ ||A_t - U V_t^T||_F^2 + \alpha \sum_{t=r-L+1}^{t-1} \theta^{r-t} \text{tr}(V_t^T L_t V) \right] + \beta \sum_{t=r-L+2} ||U_t - U_{t-1}||_F
\]

where \( \{U \in \mathbb{R}^{N \times d}, V \in \mathbb{R}^{N \times d} \} \) are latent matrices to be optimized; \( L_t \) is the Laplacian matrix of \( A_t \) with the same definition as \( L_0 \) in (13); \( \alpha > 0 \) and \( \theta \in [0, 1] \) are parameters to be specified. The first term is the standard NMF objective to learn \( \{U, V\} \) that preserve structural properties of current snapshot \( G_t \). The second **graph regularization term** [61], with a physical meaning similar to that in (13), incorporates regularization regarding previous snapshots \( G_{r-L}^{r-1} \) to \( V \), enabling \( V \) to capture evolving patterns of successive snapshots. In this setting, \( \theta \) and \( \alpha \) are used to ensure **Hypothesis 3.1** and adjust the **graph regularization term**, respectively.

Let \( \{U^*, V^*\} \) be the learned latent matrices. The decoder executes an inverse process of NMF such that

\[
\hat{A}_{r+1} = \text{Dec}(U^*, V^*) \equiv U^* V^{*T}.
\]

(5) **DeepEye.** Ahmed et al. [50] developed DeepEye that explores dynamic topology via the linear combination of multiple NMF components w.r.t. historical snapshots. Given \( G_{r-L}^r \), the encoder and loss function are defined as

\[
\text{Enc}(G_{r-L}^r) \equiv \arg\min_{U_{r-L} \geq 0, V_{r-L} \geq 0, U \geq 0, V \geq 0} \mathcal{L}(G_{r-L}^r; U_{r-L}^r, V_{r-L}^r, U, V) \\
\equiv \sum_{t=r-L+1} \theta^{r-t} \left[ ||A_t - U_t V_t||_F^2 + ||U_t - U||_F^2 + ||V_t - V||_F^2 \right]
\]

ACM Computing Surveys, Vol. 56, No. 4, Article 89. Publication date: November 2023.
where \( \{ U \in \mathbb{R}^{N \times d}, V \in \mathbb{R}^{N \times d}, U_t \in \mathbb{R}^{N \times d}, V_t \in \mathbb{R}^{N \times d} \} \) are latent matrices to be learned; \( \theta \in [0, 1] \) is the decaying factor to ensure Hypothesis 3.1. In (21), each NMF component \( \| A_t - U_t V_t^T \|_F^2 \) corresponds to a unique snapshot \( G_t \), enabling \( \{ U_t, V_t \} \) to capture structural properties of \( G_t \). \( \| U_t - U \|_F^2 \) and \( \| V_t - V \|_F^2 \) further force \( \{ U, V \} \) to comprehensively capture the dynamic structural properties w.r.t. successive snapshots \( G_{t-L} \). Let \( \{ U^*, V^*, U_t^*, V_t^* \} \) be the solution to the aforementioned objective, DeepEye has the same decoder as GrNMF defined in (20).

(6) TMF. In addition to NMF, some other related methods are based on the general matrix factorization without non-negative constraints. Yu et al. [51] formulated the dynamic topology as a function of time with learnable parameters and introduced TMF. Given historical snapshots \( G_{t-L} \), the encoder and loss function of TMF are defined as

\[
Enc(G_{t-L}^T) \equiv \arg \min_{U \in \mathbb{R}^{N \times d}}, \{ V(i) \} \mathcal{L}(G_{t-L}^T; U, \{ V(i) \})
\]

\[
\equiv \sum_{t=t-L+1}^r D_t \left\| E_t \odot \left[ A_t - U \left( \sum_{i=0}^h V(i) (t - (\tau - L))^\tau \right) \right] \right\|_F^2 + \alpha \| U \|_F^2 + \sum_{i=0}^h \beta_i \| V(i) \|_F^2,
\]

where \( \{ U \in \mathbb{R}^{N \times d}, V(i) \in \mathbb{R}^{N \times d} (0 \leq i \leq h) \} \) are model parameters to be optimized, with \( h \) as a tunable feature order; \( D_t = e^{-\theta(\tau - t)} (0 > \theta) \) is the time decaying factor to ensure Hypothesis 3.1; \( \alpha \) and \( \beta_i \) are parameters for the \( l_2 \)-regularization regarding \( \{ V(i) \} \) to avoid overfitting. \( E_t \in \mathbb{R}^{N \times N} \) is an auxiliary matrix such that \( (E_t)_{ij} = 1 \) if \( (A_t)_{ij} > 0 \) and \( (E_t)_{ij} = 0 \) otherwise. It ensures that only the pair of nodes \( (v_i^t, v_j^t) \) with an edge at time step \( t \) can contribute to the model optimization. Different from NMF-based methods (e.g., TSL, GrNMF, and DeepEye) that encode structural properties of each snapshot in one or more matrices (e.g., \( Y_t \) in \( \min_{Y_t} \sum_{i} \| A_t - Y_t V_t^T \|_F^2 \)), the model parameters \( \{ U, V(i) \} \) of TMF are shared by all time steps. In the first term, \( V_t = \sum_i V(i) [t - (\tau - L)]^\tau \) is a time-induced representation w.r.t. snapshot \( G_t \), which is also a function regarding time index \( t \). TMF is optimized via the classic gradient descent algorithm. Let \( \{ U^*, V(i)^* \} \) be the learned model parameters. The decoder of TMF is defined as

\[
\hat{A}_{t+r} = \text{Dec}(U^*, \{ V(i)^* \}) \equiv U^* \left[ \sum_{i=0}^h V(i)^*[i (\tau + r) - (\tau - L)]^\tau \right] = U^* \left[ \sum_{i=0}^h V(i)^*(L + r)^\tau \right],
\]

(23) which uses the time-induced representation \( V_{t+r} \) w.r.t. a future time step \( (\tau + r) \) to derive prediction result \( \hat{A}_{t+r} \). In this setting, TMF can support the multi-step prediction with \( 1 \leq r \leq \Delta \).

(7) LIST. Also based on the motivation of modeling dynamic topology as a function of time, Cheng et al. [52] proposed LIST that extends TMF to incorporate multi-step label propagation on each snapshot. Given historical snapshots \( G_{t-L} \), the encoder and loss function of LIST can be described as

\[
Enc(G_{t-L}^T) \equiv \arg \min_{U \in \mathbb{R}^{N \times d}}, \{ V(i) \} \mathcal{L}(G_{t-L}^T; U, \{ V(i) \})
\]

\[
\equiv \sum_{t=t-L+1}^r D_t \left\| E_t \odot (A_t - U_t V_t V_t^T) \right\|_F^2 + \sum_{i=0}^h \beta_i \| V(i) \|_F^2,
\]

where \( V_t = \sum_i V(i) [t - (\tau - L)]^\tau \) is the time-induced representation w.r.t. time step \( t \); \( \{ V(i) \in \mathbb{R}^{N \times d} (0 \leq i \leq h) \} \) are model parameters to be optimized with a tunable order \( h \); \( D_t \) and \( E_t \) are with the same definitions and physical meanings as those of TMF described in (22). Furthermore, \( U_t \equiv (1 - \hat{\theta})(I - \hat{\theta} \hat{A_t})^{-1} \) with \( \hat{A_t} = D_t^{-1/2} A_t D_t^{-1/2} \) and \( \hat{\theta} \in (0, 1) \) as a pre-set parameter is an auxiliary variable regarding the analytical solution of multi-step label propagation [62] on each snapshot \( G_t \) (see [52] for its details), which captures the high-order proximity of \( G_t \). Similar to TMF, LIST is also optimized via gradient descent. Let \( \{ V(i)^* \} \) be the learned model parameters. The decoder of LIST is defined as

\[
\hat{A}_{t+r} = \text{Dec}(\{ V(i)^* \}) \equiv V_{t+r}^T V_{t+r} = \left[ \sum_{i=0}^h V(i)^*(L + r)^\tau \right] \left[ \sum_{i=0}^h V(i)^*(L + r)^\tau \right]^T,
\]

(25)
which uses a strategy similar to that of TMF in (23) to derive the prediction result $\tilde{A}_{t+r}$ with $1 \leq r \leq \Delta$.

3.3.2 Summary of ESSD-Based OTI Methods. Compared with conventional neighbor similarity and graph summarization techniques based on manually designed heuristics, the aforementioned matrix factorization methods can automatically extract latent characteristics from dynamic topology. Some of them can also incorporate additional information (e.g., node attributes and high-order proximity) beyond the observable topology described by adjacency matrices. Moreover, all the ESSD-based OTI methods reviewed in this subsection can support TLP on weighted graphs by using adjacency matrices to describe weighted topology. However, they are still based on linear models (e.g., NMF) that cannot capture the non-linear characteristics of dynamic graphs. Due to the limitations of ESSD, they can only support coarse-grained representations of dynamic topology but may fail to handle the rapid evolution of systems. Following the OTI paradigm, they are designed and optimized for one prediction operation. Although this paradigm can capture the latest evolving patterns, when it comes to a new time step, we still need to optimize the model from scratch, which is inefficient for applications with real-time constraints. Since the dimensionality of model parameters is related to the number of nodes, the aforementioned approaches can only support the TLP in level-1 but fail to handle the variation of node sets in level-2.

3.4 ESSD-Based OTOG Methods

Most ESSD-based OTOG methods use deep learning (DL) techniques to handle dynamic topology. We categorize this type of method based on the DL module used to explore the evolving patterns across snapshots, including restricted Boltzmann machines (RBM) [63, 64], recurrent neural networks (RNNs) [65, 66], and attention mechanisms [67, 68].

3.4.1 RBM-Based Temporal Models. RBM [63, 64], with an overview depicted in Figure 6(a), is a DL structure that contains a layer of $N$ visible units $v \in \mathbb{R}^{N \times 1}$ and a layer of $d$ hidden units $h \in \mathbb{R}^{d \times 1}$, forming a fully-connected network between the two layers. $v$ and $h$ are stochastic binary units (i.e., $v_i \in \{0, 1\}$ and $h_j \in \{0, 1\}$) that encode the observable data and latent representations. In addition, there is a weight matrix $W \in \mathbb{R}^{N \times d}$ and two bias vectors $\{a \in \mathbb{R}^{N \times 1}, b \in \mathbb{R}^{d \times 1}\}$ for $\{v, h\}$. Such a structure defines a joint distribution over $h$ and $v$ that

$$P(v, h) \equiv \frac{1}{Z} \exp\{-E(v, h)\} = \frac{1}{Z} \exp[v^TWh + a^Tv + b^Th],$$

where $E(v, h) \equiv -v^TWh - a^Tv - b^Th$ is the energy function; $Z$ is a normalizing factor to ensure the normalization constraint of probability (i.e., $\sum P(v, h) = 1$); $\{W, a, b\}$ are model parameters to be optimized. One can further derive the following conditional probabilities from the joint distribution:

$$P(h_j = 1|v) = \sigma(W^T_{ij}v + b_j) \quad \text{and} \quad P(v_i = 1|h) = \sigma(W_{ij}h + a_i),$$

(Fig. 6. Overviews of the (a) standard RBM and (b) temporal RBM.)
where \( \sigma(x) = (1 + \exp(-x))^{-1} \) is the sigmoid function. The optimization of RBM aims at maximizing the likelihood (i.e., minimizing the negative log-likelihood) of training data via

\[
\min - \ln P(v) = - \ln \left( \sum_h P(v, h) \right)
\]  

(28)

(1) **ctRBM.** Li et al. [18] developed ctRBM by extending the standard RBM to handle dynamic topology. An overview of ctRBM is shown in Figure 6(b). It has two independent layers of visible units (denoted as \( v_\prec \) and \( v \)) fully connected to hidden units \( h \). \( v_\prec \) and \( v \) are used to encode historical topology \( G^T_{\tau - L} \) and prediction result \( \tilde{G}_{\tau + 1} \) (or training ground-truth \( G_{\tau + 1} \)), respectively. For each node \( v_{i, \tau} \), we set \( v_\prec = [(A_{\tau - L + 1})_{i, \cdot}] \cdot \cdots \cdot [(A_{\tau})_{i, \cdot}]^T \in \mathbb{R}^{NL \times 1} \) by concatenating the \( i \)-th rows of \( A^T_{\tau - L} \) and \( v = (\tilde{A}_{\tau + 1})^T_{i, \cdot} \in \mathbb{R}^{N \times 1} \) (or \( v = (A_{\tau + 1})^T_{i, \cdot} \)). Moreover, \( W_\prec \in \mathbb{R}^{NL \times d} \) and \( a_\prec \in \mathbb{R}^{NL \times 1} \) are the weight matrix and bias vector of the connection between \( v_\prec \) and \( h \), while \( W \in \mathbb{R}^{N \times d} \) and \( a \in \mathbb{R}^{N \times 1} \) are the weight and bias for \( v \). The encoder of ctRBM is defined as the following conditional probability:

\[
h = \text{Enc}(G^T_{\tau - L}) \equiv P(h|v, v_\prec; \delta) = \alpha \cdot \sigma(W^T_\prec v_\prec + b) + (1 - \alpha) \sigma(W^T v + b),
\]

where \( \delta = \{W, W_\prec, a_\prec, a, b\} \) denotes the set of model parameters to be optimized; \( \alpha \) is a parameter to balance components w.r.t. the historical topology and prediction result; \( h \) is the learned latent representation encoding properties of dynamic topology. In the offline training with a given training ground-truth \( A_{\tau + 1} \), we set \( v = (A_{\tau + 1})^T_{i, \cdot} \); for each node \( v_{i, \tau} \), while we let \( v = [0.5, \ldots, 0.5]^T \) (i.e., a constant vector with all elements set to 0.5) for the online generalization without available ground-truth. Given \( h \), the decoder of ctRBM is then defined as

\[
(\tilde{A}_{\tau + 1})^T_{i, \cdot} = \text{Dec}(h) \equiv P(v|h; \delta) = \sigma(W h + a),
\]

which derives the \( i \)-th row of the prediction result \( \tilde{A}_{\tau + 1} \) for node \( v_{i, \tau} \).

Note that the encoder and decoder are defined for each node \( v_{i, \tau} \in V \), where \( V \) is assumed to be fixed for all the snapshots. Li et al. [18] suggested training a ctRBM model for each node. Due to the OTOG paradigm, the ground-truth \( G_{\tau + 1} \) (in terms of \( A_{\tau + 1} \)) w.r.t. the snapshot to be predicted is given in the offline training. The loss function w.r.t. the prediction of a node \( v_{i, \tau} \) is defined as

\[
\min_\delta L(G^T_{\tau - L}, G_{\tau + 1}, v_{i, \tau}; \delta) \equiv - \ln P(v = (A_{\tau + 1})_{i, \cdot}; \delta) = - \ln \sum_{h, v_\prec} P(v = (A_{\tau + 1})_{i, \cdot}, v_\prec, h; \delta).
\]

In most cases, directly optimizing this objective is intractable. Contrastive divergence [69], an alternative approximated algorithm, is adopted to optimize ctRBM. Compared with conventional linear models (e.g., matrix factorization introduced in Section 3.3.1), RBM-based methods can capture non-linear characteristics of dynamic topology. However, to enable RBM to handle dynamic topology with successive snapshots, we concatenate rows of historical adjacency matrices to a long vector. In this setting, the dimensionality of model parameters is related to the number of nodes. Hence, RBM-based approaches can only tackle the TLP in level-1, where all the snapshots share a common node set. Such a setting of RBM also fails to utilize the sparsity of graph topology and usually has high complexity of model parameters.

3.4.2 **RNN-Based Temporal Models.** Most ESSD-based OTOG approaches that use RNNs to capture temporal characteristics, can be described by the framework depicted in Figure 7. It is an extension of our encoder-decoder framework introduced in Section 2.5, where the encoder is further divided into (i) a **structural encoder** and (ii) a **temporal encoder**.

For each snapshot \( G_t \), the **structural encoder** maps adjacency matrix \( A_t \) and attribute matrix \( X_t \) (if available) to latent embedding \( Z_t \in \mathbb{R}^{N \times d} \), capturing the structural properties of each single snapshot \( G_t \). The **temporal encoder** then maps \( Z^T_{\tau - L} \) w.r.t. \( G^T_{\tau - L} \) to another representation \( Z \in \mathbb{R}^{N \times d} \), and usually has high complexity of model parameters.
\( \mathbb{R}^{N \times d} \), capturing the evolving patterns across successive snapshots. In this setting, the \( i \)th rows of \( Z_t \) and \( Z \) can be the embedding of node \( v_i \). Finally, the decoder takes \( Z \) as input and derives prediction result \( \hat{A}_{t+1} \) (i.e., mapping embedding \( Z_t \) of \( v_i \) to the \( i \)th row of \( \hat{A}_{t+1} \)). In addition, some methods first reshape \( Z_t \) to a row-wise long vector \( z_t \in \mathbb{R}^{N \times 1} \) before feeding it to the temporal encoder and then obtain another long vector \( z \in \mathbb{R}^{N \times 1} \) from the temporal encoder, where \( z_t \) and \( z \) can be considered as snapshot-level embedding. The decoder takes \( z \) as input and outputs another long vector \( \hat{a}_{t+1} \in \mathbb{R}^{N \times 1} \), which is further reshaped to a matrix form \( \hat{A}_{t+1} \in \mathbb{R}^{N \times N} \) as the prediction result. Some approaches also adopt a stacked multi-layer structure for their encoders, with each layer containing a structural encoder and a temporal encoder.

Some RNN structures (e.g., long short-term memory (LSTM) [65] and gated recurrent unit (GRU) [66]) can be used to build the temporal encoders of an ESSS-based OTOG approach.

1) Dyngraph2vec. Based on the framework in Figure 7, Goyal et al. [27] proposed dyngraph2vec with three variants. One variant uses multi-layer perceptron (MLP) and LSTM to build its structural and temporal encoders. Due to space limit, we leave details of MLP and LSTM in supplementary materials. The encoder of dyngraph2vec can be described as

\[
Z = \text{Enc}(G_{t-L}^r) \equiv \text{LSTM}(Z_{t-L}^r)_{L} \quad \text{with} \quad Z_t \equiv \text{MLP}(A_t) \quad (r - L < t \leq r). \tag{32}
\]

In (32), the temporal encoder (i.e., LSTM) in sequence takes \( Z_{t-L}^r \) as input and then outputs a series of hidden states \([H_1, \ldots, H_N] = \text{LSTM}(Z_{t-L}^r)_{L}\). Only the last state \( H_L \) is adopted as the final output of the temporal encoder denoted as \( Z = H_L = \text{LSTM}(Z_{t-L}^r)_{L} \). The decoder further uses another MLP to map \( Z \) to the predicted adjacency matrix via

\[
\hat{A}_{t+1} = \text{Dec}(Z) \equiv \text{MLP}(Z). \tag{33}
\]

For one prediction operation with \( G_{t+1} \) as the ground-truth, the loss function of dyngraph2vec is defined as

\[
\min_\delta \mathcal{L}(G_{t-L}^r, G_{t+1}; \delta) \equiv \left\| (\hat{A}_{t+1} - A_{t+1}) \odot E_{t+1} \right\|^2_F, \tag{34}
\]

where \( E_{t+1} \) is an auxiliary variable to give different penalties to the observed and non-existent edges in \( G_{t+1} \). We set \((E_{t+1})_{ij} = \beta \) if \((A_{t+1})_{ij} > 0 \) and \( E_{t+1} = I \) otherwise, with \( \beta > 0 \) as a tunable parameter. Chen et al. [70] proposed E-LSTM-D based on the same encoder, decoder, and loss function as the aforementioned variant of dyngraph2vec.

2) DDNE. Li et al. [26] introduced DDNE, which does not specify its structural encoder but directly uses adjacency matrices as snapshot-induced features (i.e., \( Z_t = A_t \)). The temporal encoder contains two GRUs with different directions regarding the sequence \( Z_{t-L}^r \equiv A_{t-L}^r \). Due to space limit, we leave details of GRU in supplementary materials. For simplicity, the encoder of DDNE can be described as

\[
Z = \text{Enc}(G_{t-L}^r) \equiv \left[ \text{GRU}^{-r}(A_{t-L}^r) || \text{GRU}^{+r}(A_{t-L}^r) \right]. \tag{35}
\]
The first GRU in sequence takes \( A_{τ−L+1}, \ldots, A_τ \) as input and derives states \( [H^+_1, \ldots, H^-_L] \). In contrast, the input and output of the second GRU are denoted as \( [A_τ, \ldots, A_{τ−L+1}] \) and \( [H^+_1, \ldots, H^-_L] \). The output of the temporal encoder is rearranged as \( Z = [H^+_1||H^+_2||\cdots||H^+_L||H^-_1||H^-_2||\cdots||H^-_L] \).

Given \( Z \), the decoder of DDNE has the same definition as dyngraph2vec in (33). The original version of DDNE considers TLP on unweighted graphs, treating it as a binary edge classification task. The loss function w.r.t. one prediction operation combines the classic cross-entropy loss with graph regularization regarding historical connection frequency, which is defined as

\[
\min_δ \mathcal{L}(G^*_{τ-L}, G_{τ+1}; δ) \equiv \sum_{i,j=1}^N (E_{τ+1})_{ij} (A_{τ+1})_{ij} \ln (\tilde{A}_{τ+1})_{ij} + α tr(Z^T \mathbf{L}_N Z),
\]

where \( E_{τ+1} \) is an auxiliary variable for the first term (i.e., cross-entropy loss) with the same definition and physical meaning as that in (34); \( \mathbf{L}_N = \mathbf{D}_N - \mathbf{N} \) is the Laplacian matrix of an auxiliary matrix \( \mathbf{N} = \sum_{τ=τ−L+1}^τ \mathbf{A}_τ \); \( α \) is a tunable parameter. Since DDNE assumes \( (A_τ)_{ij} \in \{0, 1\} \), it is used to describe unweighted topology, \( \mathbf{N}_{ij} \) is the historical connection frequency between \( (v_i, v_j) \). The second term of graph regularization can be rewritten as \( \text{tr}(Z^T \mathbf{L}_N Z) = 0.5 \mathbf{N}_{ij} [Z_{t_1} - Z_{t_2}]^2 \), which regularizes the embedding \( Z \) using connection frequency \( \mathbf{N} \), with a physical meaning similar to that in (13).

As DDNE is a typical task-dependent DNE-based method, it can also be extended to support the prediction of weighted topology by replacing the loss function (36) with that of dyngraph2vec defined in (34).

(3) **EvolveGCN.** Instead of MLP, EvolveGCN [53] uses GCN [71], a type of GNN, to build its structural encoder. RNN is adopted as the temporal encoder to evolve parameters of GCN rather than handling snapshot-induced embedding \( Z^{τ}_{t−L} \) like dyngraph2vec. We leave details of GCN and RNN (e.g., LSTM and GRU) in supplementary materials. The encoder of EvolveGCN is a multi-layer structure, with each layer containing a structural encoder and a temporal encoder.

Two variants of EvolveGCN with two manners to evolve model parameters of GCN were proposed in [53]. Let \( Z^{(k−1)}_t \) and \( Z^{(k)}_t \) be the input and output of the \( k \)th encoder layer w.r.t. each snapshot \( G_t \), where \( Z^{(0)}_t \equiv X_t \) (i.e., available node attributes in \( G_t \)). Let \( W^{(k−1)}_t \) be the weight (i.e., learnable model parameters) of GCN in the \( k \)th layer at time step \( t \). The \( k \)th encoder layer of the first variant can be described as

\[
Z^{(k)}_t = \text{Enc}^{(k)}(G_t) \equiv \text{GCN}(A_t, Z^{(k−1)}_t, W^{(k−1)}_t), \quad \text{with} \quad W^{(k−1)}_t = \text{GRU}(Z^{(k−1)}_t, W^{(k−1)}_{t−1}).
\]

In each time step \( t \), the \( k \)th encoder layer derives a new GCN weight \( W^{(k−1)}_t \) by letting \( Z^{(k−1)}_t \) and \( W^{(k−1)}_t \) be the feature input and the previous hidden state of GRU. The \( k \)th encoder layer of the second variant is defined as

\[
Z^{(k)}_t = \text{Enc}^{(k)}(G_t) \equiv \text{GCN}(A_t, Z^{(k−1)}_t, W^{(k−1)}_t), \quad \text{with} \quad W^{(k−1)}_t = \text{LSTM}(W^{(k−1)}_{t−1}, W^{(k−1)}_{t−1}).
\]

which updates GCN weight \( W^{(k−1)}_t \) by letting \( W^{(k−1)}_{t−1} \) be both the feature input and previous hidden state of LSTM. The encoder adopts the GCN output of the last encoder layer w.r.t. current time step \( τ \) as the temporal embedding \( Z \).

**EvolveGCN** is a task-dependent DNE method. One should specify the decoder and training loss related to the downstream task (i.e., TLP). Pareja et al. [53] recommended using the following decoder

\[
(\tilde{A}_{τ+1})_{ij} = \text{Dec}(Z) \equiv \text{MLP}([Z_{t_1}||Z_{t_2}]).
\]

It concatenates embedding \( (Z_{t_1}, Z_{t_2}) \) w.r.t. each node pair \( (v_i, v_j) \) and applies an MLP to map \( [Z_{t_1}||Z_{t_2}] \) to \( (\tilde{A}_{τ+1})_{ij} \). The original version of EvolveGCN only considers the TLP on unweighted
graphs. One can use the following binary cross-entropy loss w.r.t. one prediction operation to train the model in an end-to-end manner:

\[
\min_{\delta} \mathcal{L}(G^\tau_{t-L}, G_{t+1}; \delta) = \sum_{i,j=1}^{N_t} -\left[(A_{t+1})_{ij} \ln(\hat{A}_{t+1})_{ij} + (1 - (A_{t+1})_{ij}) \ln(1 - (\hat{A}_{t+1})_{ij})\right]/N_t.
\]

(40)

To extend EvolveGCN to support the prediction of weighted topology, we can replace (40) with the loss function of dyngraph2vec defined in (34).

4. GCN-GAN. Most existing methods merely consider inference tasks on unweighted graphs, while the TLP on weighted graphs is seldom studied. Some approaches (e.g., \textit{neighbor similarity} and ctRBM) may even fail to capture and predict weighted topology. Although several methods (e.g., GrNMF, DeepEye, DDNE, and dyngraph2vec) can still tackle weighted TLP, they can only derive low-quality prediction results. We elaborate on this advanced topic regarding the TLP on weighted graphs later in Section 4.1.

Inspired by the high-resolution video prediction [72] using generative adversarial network (GAN) [73, 74], Lei et al. [7] focused on the weighted TLP and proposed GCN-GAN. It combines the extended framework in Figure 7 with GAN and can derive high-quality prediction results for weighted graphs. Following GAN, the model contains a generator \( \mathcal{G} \) and a discriminator \( \mathcal{D} \). \( \mathcal{G} \) adopts the encoder-decoder framework to generate prediction results while \( \mathcal{D} \) is an auxiliary structure to refine the generated results. In addition to the historical topology described by \( A^\tau_{t-L} \), GCN-GAN also generates random noise via \( U[0, 1] \) to support GAN (i.e., generating plausible samples from noise), which are treated as attribute inputs described by \( X^{t}_t \). Given \( A^\tau_{t-L} \) and \( X^{t}_t \) w.r.t. \( G^\tau_{t-L} \), the encoder of GCN-GAN is defined as

\[
z = \text{Enc}(G^\tau_{t-L}) \equiv \text{LSTM}(z^{t}_{t-L})_L \text{ with } z_t \equiv r_>(Z_t) \text{ and } Z_t \equiv \text{GCN}(A_t, X_t) \text{ (} \tau - L < t \leq \tau \),
\]

(41)

where GCN and LSTM (see supplementary materials for their details) are used to build the \textit{structural} and \textit{temporal encoders}. For each snapshot \( G_t \), GCN takes \( A_t \) and \( X_t \) as inputs and derives embedding \( Z_t \equiv \mathcal{N}^{N \times d} \). A function \( r_>(\cdot) \) is then applied to reshape \( Z_t \) to a row-wise long vector \( z_t \equiv \mathcal{R}^{N \times d} \) (i.e., second strategy in Figure 7) before feeding it to the LSTM. In this setting, the final output of the encoder is also a vector \( z \equiv \mathcal{R}^{N \times d} \) (i.e., the last hidden state of LSTM), preserving the evolving patterns of successive snapshots. Given \( z \), the decoder of GCN-GAN is defined as

\[
\hat{A}_{t+1} = \text{Dec}(z) \equiv r_<(\hat{A}_{t+1}) \text{ with } \hat{A}_{t+1} \equiv \text{MLP}(z).
\]

(42)

It uses an MLP to map \( z \) to a row-wise long vector \( \hat{A}_{t+1} \equiv \mathcal{R}^{N \times 1} \). Another function \( r_<(\cdot) \) is applied to reshape \( \hat{A}_{t+1} \) to the matrix form \( \hat{A}_{t+1} \equiv \mathcal{R}^{N \times N} \) as the prediction result. In addition to \( \mathcal{G} \) with an encoder and a decoder, \( \mathcal{D} \) is an auxiliary structure defined as

\[
p = \mathcal{D}(M) \equiv \text{MLP}(m) \text{ with } m \equiv r_>(M),
\]

(43)

where \( M \equiv \{A_{t+1}, \hat{A}_{t+1}\} \); \( r_>(\cdot) \) is a function to reshape \( M \equiv \mathcal{R}^{N \times N} \) to a row-wise long vector \( m \equiv \mathcal{R}^{N \times 1} \); \( p \) denotes the probability that \( M = A_{t+1} \) rather than \( M = \hat{A}_{t+1} \).

Let \( \delta_D \) and \( \delta_G \) be sets of model parameters in \( \mathcal{D} \) and \( \mathcal{G} \). The optimization of GCN-GAN includes the (i) pre-training of \( \mathcal{G} \) and (ii) joint optimization of \( \mathcal{D} \) and \( \mathcal{G} \). The pre-training loss of \( \mathcal{G} \) is with the same definition as (34). After pre-training, the model can preliminarily generate predicted snapshot \( \hat{A}_{t+1} \) consistent with ground-truth \( A_{t+1} \). In formal optimization, GCN-GAN adopts the losses of GAN. On the one hand, \( \mathcal{D} \) tries to distinguish \( A_{t+1} \) from \( \hat{A}_{t+1} \) via

\[
\min_{\delta_D} \mathcal{L}_D(G^\tau_{t-L}, G_{t+1}; \delta_D) \equiv -[\ln(1 - \mathcal{D}(A_{t+1})) + \ln \mathcal{D}(A_{t+1})].
\]

(44)

On the other hand, \( \mathcal{G} \) tries to generate a plausible snapshot \( \hat{A}_{t+1} \) to fool \( \mathcal{D} \) via

\[
\min_{\delta_G} \mathcal{L}_G(G^\tau_{t-L}, G_{t+1}; \delta_G) \equiv -\ln \mathcal{D}(\hat{A}_{t+1}).
\]

(45)
During the joint optimization between $D$ and $G$, $\hat{A}_{r+1}$ can be further refined and is expected to be a high-quality prediction result for weighted graphs. As the model parameters of RNN in the encoder and MLP in the decoder are related to the number of nodes, GCN-GAN can only support TLP in level-1, failing to tackle the variation of nodes.

5) **IDEA.** Qin et al. [44] introduced IDEA that extends GCN-GAN to the weighted TLP in level-2. Similar to GCN-GAN, IDEA also contains i) a generator $G$ following the encoder-decoder framework and ii) a discriminator $D$.

The encoder in $G$ is a multi-layer structure, with the GCN and a modified GRU (see supplementary materials for details of GCN and GRU) used to build the structural and temporal encoders of each layer. In addition to adjacency matrices $A^r_{t-L}$ and attribute matrices $X^r_{t-L}$ that describe topology and attributes of $G^r_{t-L}$, IDEA also maintains an aligning matrix $B_t \in \mathbb{R}^{N_t \times N_{t+1}}$ for successive snapshots $\{G_t, G_{t+1}\}$ to encode the variation of node sets. $(B_t)_{ij} = 1$ if $v^t_i$ corresponds to $v^{t+1}_j$ and $(B_t)_{ij} = 0$ otherwise. Let $Z^{(k-1)}_t$ and $Z^{(k)}_t$ be the input and output of the $k$th encoder layer at time step $t$, where $Z_t^{(0)} \equiv$ MLP($X_t$). For each time step $t$, the $k$th encoder layer takes adjacency matrix $A_t$, attribute matrices $\{X_{t-1}, X_t\}$, aligning matrix $B_t$, and previous encoder output $Z^{(k-1)}_t \in \mathbb{R}^{N_{t-1} \times d}$ (in the same layer) as the joint inputs, and then derives embedding $Z^{(k)}_t \in \mathbb{R}^{N_t \times d}$, which can be described as

$$Z^{(k)}_t = \text{Enc}^{(k)}(A_t, X_{t-1}, X_t, B_t, Z^{(k-1)}_t) \equiv \text{GRU}(G^{(k)}_t, Z^{(k-1)}_t),$$

with $G^{(k)}_t \equiv \text{GCN}(A_t, Z^{(k-1)}_t)$

$$Z^{(1)}_{t-1} = \left[ B_{t-1} + a(X_{t-1}, X_t) \right]^{T} Z^{(1)}_{t-1}, \text{ and } a(X_{t-1}, X_t) \equiv \text{MLP}(X_{t-1})\text{MLP}(X_t)^T.$$ (46)

In (46), we first obtain auxiliary embedding $G^{(k)}_t \equiv \text{GCN}(A_t, Z^{(k-1)}_t) \in \mathbb{R}^{N_{t-1} \times d}$ that preserves structural properties of snapshot $G_t$. Before feeding $G^{(k)}_t$ and the hidden state $Z^{(k)}_{t-1} \in \mathbb{R}^{N_{t-1} \times d}$ that matches with the node set $V_{t-1}$ of $G_{t-1}$ to GRU, we derive the aligned state $\hat{Z}^{(k)}_{t-1} \equiv [B_{t-1} + a(X_{t-1}, X_t)]^{T} Z^{(k)}_{t-1} \in \mathbb{R}^{N_t \times d}$ that matches with $V_t$ by mapping the rows of $Z^{(k)}_{t-1}$ to those of $\hat{Z}^{(k)}_{t-1}$. In addition to the node correspondence encoded in $B_t$, an attentive aligning unit $a(X_{t-1}, X_t)$ is introduced to extract additional aligning relations from attributes $\{X_{t-1}, X_t\}$. In this setting, the output $\hat{Z}^{(k)}_t \in \mathbb{R}^{N_t \times d}$ can match with $V_t$ to handle the variation of node sets (e.g., $V_{t-1} \neq V_t$) while preserving the evolving patterns across snapshots.

For current time step $t$, let the aligned embedding $\hat{Z}_t \in \mathbb{R}^{N_{t+1} \times d}$ be the final output of the encoder, which matches with $V_{t+1}$. Different from dyngraph2vec and DDNE that directly map the derived embedding to the prediction result $\hat{A}_{t+1}$ via an MLP in (33), the decoder of IDEA use the aggregation of $\hat{Z}_t$ to generate each element $(\hat{A}_{t+1})_{ij}$ via

$$(\hat{A}_{t+1})_{ij} = \text{Dec}(\hat{Z}_t) \equiv 1 + \text{tanh}(-\xi_{ij})(U_t)_{i.} - (U_t)_{.j},$$ with $\xi_{ij} \equiv (V_t V_t^T)_{ij}$, $V_t \equiv \text{MLP}(\hat{Z}_t)$, and $U_t \equiv \text{MLP}(\hat{Z}_t)$.

$D$ is an auxiliary structure defined as $p = D(M, X_{r+1}) \equiv \text{MLP}(\text{GCN}(M, X_{r+1}))$, where $M \in \{A_{r+1}, \hat{A}_{r+1}\}$; $p$ is an $N_{r+1}$-dimensional vector with $p_i$ as the probability that $M_{i.} = (A_{r+1})_{i.}$ rather than $(\hat{A}_{r+1})_{i.}$. Given historical snapshots $G^r_{t-L}$, IDEA can in sequence generate prediction results $\hat{A}^{r+1}_{t-L+1}$ w.r.t. $G^r_{t-L+1}$. All the results in $\hat{A}^{r+1}_{t-L+1}$ are used for model optimization. In particular, IDEA combines the adversarial learning loss $\mathcal{L}_{\text{AL}}$ of GAN with the conventional error minimization loss $\mathcal{L}_{\text{EM}}$ and a novel scale difference minimization loss $\mathcal{L}_{\text{SDM}}$ to optimize $G$, which are defined as

$$\mathcal{L}_{\text{AL}}(G_t) \equiv - \sum_i \ln D(A_t, X_t)/N_t, \mathcal{L}_{\text{EM}}(G_t) \equiv \|A_t - \hat{A}_t\|_F^2 + \sum_j |(A_t)_{ij} - (\hat{A}_t)_{ij}|,$$

and $\mathcal{L}_{\text{SDM}}(G_t) \equiv \sum_j \|\log_{10}[(\hat{U}_t)_{ij}]/(\bar{V}_t)_{ij}]\|$ for each snapshot $G_t$. $\mathcal{L}_{\text{EM}}$ minimizes the reconstruction errors measured by the F-norm and l1-norm to help $G$ derive prediction result $\hat{A}_t$ consistent with ground-truth $A_t$. In addition to $\mathcal{L}_{\text{AL}}, \mathcal{L}_{\text{SDM}}$ can also refine the generated prediction results by using $\log_{10}(\cdot)$ to
minimize the scale difference between \( \{A_t, \tilde{A}\} \), where \( \{\tilde{U}_t, \tilde{V}_t\} \) are applied to clip \( \{A_t, \tilde{A}\} \) with the same motivations and definitions as the MLSD metric (see Section 2.3 and supplementary materials for its details). The loss functions to optimize \( G \) and \( D \) are then defined as

\[
\min_{\delta_G} \mathcal{L}_G \left( G_{\tau-L+1}^{\tau+1} ; \delta_G \right) = \sum_{t=\tau-L+1}^{\tau+1} D_t \left[ \mathcal{L}_{AL}(G_t) + \alpha \mathcal{L}_{EM}(G_t) + \beta \mathcal{L}_{SDM}(G_t) \right],
\]

\[
\min_{\delta_D} \mathcal{L}_D \left( G_{\tau-L+1}^{\tau+1} ; \delta_D \right) = -\sum_{t=\tau-L+1}^{\tau+1} D_t \sum_i \left[ \ln(1 - \mathcal{D}(\tilde{A}_t, X_t)) + \ln \mathcal{D}(A_t, X_t) \right] / N_t,
\]

where \( D_t \equiv (1 - \theta)^{r+1-t} \) (with \( \theta \in [0, 1] \) as a tunable parameter) is the decaying factor integrating

**Hypothesis 3.1:** \( \alpha \) and \( \beta \) are pre-set parameters to adjust the contributions of \( \mathcal{L}_{EM} \) and \( \mathcal{L}_{SDM} \).

Compared with RBM-based approaches (e.g., ctRBM) that concatenate historical adjacency matrices, the aforementioned RNN-based methods, with model parameters shared by successive time steps, are more space-efficient to handle dynamic topology. As the dimensionality of the temporal encoders (i.e., RNN) and decoders (i.e., MLP of dyngraph2vec, DDNE, and GCN-GAN) is related to the number of nodes \( N \), these approaches can only deal with the TLP in level-1, assuming that all the snapshots share a common node set. In contrast, RNN in EvolveGCN is used to evolve parameters of GNN that are not related to \( N \). IDEA adopts a modified RNN that aligns the non-fixed node sets between successive snapshots. The decoders of EvolveGCN and IDEA are also not related to \( N \). Therefore, EvolveGCN and IDEA can support level-2 and handle the variation of node sets.

### 3.4.3 Attention-Based Temporal Models

Most existing ESSD-based OTOG methods, which use attention mechanisms \([67, 68]\) to capture evolving patterns of successive snapshots, also follow the extended encoder-decoder framework in Figure 7, with attention as building blocks of the temporal encoder. Due to space limitation, we elaborate on the general form of attention in supplementary materials.

1. **STGSN.** Min et al. \([28]\) proposed STGSN using GCN and attention to build the structural and temporal encoders. In addition to the historical topology described by \( A_{t-L}^\tau \), an auxiliary adjacency matrix \( A_{gbl}^\tau \equiv \sum_{t=\tau-L+1}^{\tau+1} A_t \) is introduced to encode the “global” topology of successive snapshots. Node attributes described by \( X_{t-L}^\tau \) are also assumed to be available. Given the (i) historical topology \( A_{t-L}^\tau \) and (ii) historical attributes \( X_{t-L}^\tau \), (iii) “global” topology \( A_{gbl}^\tau \), and (iv) attributes \( X_{t+1}^\tau \) of the next snapshot, the encoder that derives the embedding \( Z_{i:t} \), for each node \( v_i \), is defined as

\[
Z_{i:t} = \text{Enc}(G_{t-L}^\tau, X_{t+1}^\tau) = [\text{Att}(q, K, V)](Z_{gbl}^\tau)_{i:t}, \quad Z_{gbl} = \text{GCN}(A_{gbl}, X_{t+1}^\tau),
\]

\[
K_{i:t} = V_{i:t} = (Z_{i:t}), \quad \text{and} \quad Z_{t+1} = \text{GCN}(A_{t+1}, X_{t+1}), \quad (\tau - L < t \leq \tau).
\]

We first derive the snapshot-induced embedding \( Z_t = \text{GCN}(A_t, X_t) \) \((\tau - L < t \leq \tau)\) and auxiliary “global” embedding \( Z_{gbl} = \text{GCN}(A_{gbl}, X_{t+1}^\tau) \) via GCN. An attention unit \( \text{Att}(q, K, V) \) is then applied to capture evolving patterns across snapshots, where \( q \in \mathbb{R}^{1 \times d}, K \in \mathbb{R}^{L \times d}, \) and \( V \in \mathbb{R}^{L \times d} \) are inputs of query, key, and value. We leave details of GCN and attention in supplementary materials. For each node \( v_i \), we let \( q \) be the global embedding of \( v_i \) (i.e., \( q = (Z_{gbl})_{i:t} \)). \( K \) and \( V \) are set to be the concatenation of snapshot-induced embedding \( \{Z_t\} \) of \( v_i \) over historical snapshots \( G_{t-L}^\tau \), with the \( t \)-th row as the embedding w.r.t. time step \( t \) (i.e., \( K_{i:t} = V_{i:t} = (Z_{i:t}) \)). Accordingly, the output of \( \text{Att}(q, K, V) \) is a vector associated with \( v_i \). The encoder treats the concatenated vector \( Z_{i:t} = [\text{Att}(q, K, V)](Z_{gbl})_{i:t} \) as the final embedding output of \( v_i \). As STGSN is a task-dependent DNE method, one can use the same decoder and training loss as EvolveGCN defined in (39), (40), and (34).

2. **DySAT.** Sankar et al. \([30]\) developed DySAT that adopts GAT \([75]\), a type of GNN, and self-attention as building blocks of the structural and temporal encoders. Given historical topology \( A_{t-L}^\tau \) and attributes \( X_{t-L}^\tau \), the encoder of DySAT derives corresponding embedding \( Z_{i:t} \) for each node
\( \tau \) via the following procedure:

\[
Z_{i:} = \text{Enc}(G^{t}_{r-L}) \equiv r_{>}(\text{Att}(Q, K, V)) \quad \text{with} \quad Q = K = V = \left[ (Z^{t}_{r-L+1})_{i:} \ldots \| (Z^{t}_{r})_{i:} \right]^{T}
\]

and \( Z_{t} \equiv \text{GAT}(A_{t}, X_{t}) \) \((\tau - L < t \leq \tau)\).

The structural encoder first generates the snapshot-induced embedding \( Z_{t} \equiv \text{GAT}(A_{t}, X_{t}) \) for each historical snapshot \( G_{t} \), using GAT (see supplementary materials for its details). For each node \( v_{i} \), the temporal encoder applies attention to derive embedding \( Z_{i:} \) that preserves the evolving patterns across snapshots, where the query, key, and value are set to be the concatenation of snapshot-induced embedding of \( v_{i} \) over \( G^{r}_{r-L} \) (i.e., \( Q = K = V = \left[ (Z^{t}_{r-L+1})_{i:} \ldots \| (Z^{t}_{r})_{i:} \right]^{T} \in \mathbb{R}^{L \times d} \)). Accordingly, the output of \( \text{Att}(Q, K, V) \) is also an \( L \times d \) matrix. Another function \( r_{>}(\cdot) \) is introduced to reshape \( \text{Att}(Q, K, V) \) to a row-wise long vector \( Z_{i:} \in \mathbb{R}^{Ld \times 1} \) as the dynamic embedding of \( v_{i} \).

DysAT is a task-independent DNE approach. In [30], random walks on each snapshot were used to optimize the model. Let \( W_{t}^{i} \) and \( P_{t}^{i} \) be the sets of (i) nodes that co-occur with \( v_{i} \) in fixed-length random walks and (ii) negative samples of \( v_{i} \) on \( G_{t} \). The loss of DysAT maximizes the likelihood (i.e., minimizing the negative log-likelihood) formulated by embedding \( Z \) w.r.t. the sampled random walks. The approximated loss with negative sampling is defined as

\[
\min_{\delta} L(G^{r}_{r-L}; \delta) \equiv \sum_{t=r-L+1}^{r} \sum_{v_{i} \in V} \left[ \sum_{v_{j} \in W_{t}^{i}} \sum_{v_{k} \in P_{t}^{i}} -\ln \sigma(Z_{i:}, Z_{j:}^{T}) - n_{s} \sum_{v_{k} \in P_{t}^{i}} \ln(1 - \sigma(Z_{i:}, Z_{k:}^{T})) \right],
\]

where \( n_{s} \) is the number of negative samples w.r.t. each node on a snapshot; \( \sigma(\cdot) \) is the sigmoid function. To derive the prediction result, one can adopt one of the strategies illustrated in Table 1 to build the decoder of DysAT.

Different from some RNN-based methods with the dimensionality of model parameters related to the number of nodes (e.g., dynagraph2vec, DDNE, and GCN-GAN), the aforementioned attention-based approaches (i.e., STGSN and DysAT) can be generalized to new unseen nodes and handle the variation of node sets (i.e., TLP in level-2), based on the inductive nature of GNNs [76, 77] and attentive combination of snapshot-induced embedding \( Z^{r}_{r-L} \).

### 3.4.4 Summary of ESSD-Based OTOG Methods

Compared with conventional linear models (e.g., neighbor similarity, graph summarization, and matrix factorization introduced in Section 3.3), the aforementioned ESSD-based OTOG methods can explore the non-linear characteristics of dynamic graphs via DL structures (e.g., RBM, MLP, GNN, RNN, and attention). Following the OTOG paradigm, these methods have the potential to satisfy the real-time constraints of systems, because there is no additional optimization in online generalization. Most of the methods (except cTRBM) can directly (or be easily adapted to) support the TLP on weighted graphs by using adjacency matrices to describe weighted topologies. Some approaches (e.g., GCN-GAN and IDEA) can even derive high-quality weighted prediction results. However, they may suffer from the limitations of ESSD which can only support the coarse-grained representations of dynamic graphs. The adopted OTOG paradigm may also have the risk of failing to capture the latest evolution of dynamic graphs in online generalization.

### 3.5 UESD-Based OTI Methods

Existing UESD-based OTI approaches usually follow the embedding lookup scheme of classic network embedding techniques [78, 79]. In this scheme, there is an embedding lookup table \( Z \in \mathbb{R}^{N \times d} \) shared by all the time steps, with \( Z_{i:} \), mapping node \( v_{i} \) to its embedding. \( Z \) is also the model parameter to be optimized, whose dimensionality is related to the number of nodes \( N \), and thus can only be used to support the TLP in level-1. In general, the encoder of this type of method can be

ACM Computing Surveys, Vol. 56, No. 4, Article 89. Publication date: November 2023.
described as

$$\text{Enc}(G_t) \equiv Z \in \mathbb{R}^{N \times d}.$$ 

(53)

We divide related methods into two categories according to their techniques used to capture the evolving patterns of UESD-based topology, which are temporal random walk (TRW) and temporal point process (TPP).

3.5.1 Temporal Random Walk (TRW). Inspired by the random walk on static graphs [78, 79], TRW is an extension to dynamic graphs with UESD. A TRW with length \( K \) can be defined as \( \omega = (v^{(0)}, v^{(1)}, \ldots, v^{(K)}) \) such that \( v^{(r)} \in V_t \), \((v^{(r-1)}, v^{(r)}) \in E_t \), and \( t^{(r)} \in \Gamma \). To enable TRWs to capture the evolution of topology, we also assume that \( t^{(r-1)} \leq t^{(r)} \). Namely, each TRW is sampled in ascending order of time steps. For the example in Figure 8(a), \((v_1, v_4, v_5, v_2, v_3)\) is a valid TRW but \((v_1, v_4, v_3, v_2, v_5)\) is not valid because \( t^{(1)} = t_4 > t^{(2)} = t_3 \).

1) **CTDNE.** Based on TRW, Nguyen et al. [19] introduced CTDNE following the embedding lookup scheme. The encoder of CTDNE is already defined in (53). Let \( W \) be the set of sampled TRWs. The loss function can be described as

$$\min_{Z} \mathcal{L}(W, Z) = -\sum_{\omega \in W} \sum_{v_i \in \omega} \sum_{\omega \in \omega \setminus \{v_i\}} \ln P(v_j | v_i; Z),$$

with

$$P(v_j | v_i; Z) = \frac{\exp\{Z_{j,k}^{\top}Z_{i,k}\}}{\sum_{v_k \in V_t} \exp\{Z_{k,k}^{\top}Z_{i,k}\}},$$

(54)

which maximizes the likelihood (i.e., minimizing the negative log-likelihood) of each TRW \( \omega \in \mathcal{W} \). Given a node \( v_i \) selected in a TRW \( \omega \), it maximizes the co-occurrence probability \( P(v_j | v_i; Z) \) for each rest node \( v_j \in \omega \setminus \{v_i\} \), which is derived by the softmax of embedding w.r.t. associated nodes. Directly computing \( P(v_j | v_i; Z) \) via softmax is usually time-consuming due to the summation of all nodes in the denominator. Instead, negative sampling [80] can be used to derive an approximated loss similar to that of DySAT in (52). As CTDNE is a task-independent DNE method, we can use one of the strategies in Table 1 to define its decoder.

3.5.2 Temporal Point Processes (TPP). TPP is a continuous-time stochastic process that can also be used to formulate the UESD-based dynamic topology. Assuming that an event happens in a tiny period \([t, t + dt]\), TPP represents the conditional probability of this event given historical events as \( \lambda(t)dt \). Hawkes process [38] is a typical TPP with a conditional intensity defined as

$$\lambda(t) = \mu(t) + \int_{-\infty}^{t} \kappa(t - s)dn(s),$$

(55)

where \( \lambda(t) \) is the conditional intensity; \( \mu(t) \) is the base intensity describing the arrival rate of a spontaneous event at time \( t \); \( \kappa(t - s) \) is the kernel modeling the time decay of past events; \( n(t) \) denotes the number of events until \( t \). Methods based on the Hawkes process usually use dynamic embedding \( Z \) to formulate \( (\lambda(t), \mu(t), \kappa(t - s), n(t)) \).

1) **HTNE.** Zuo et al. [54] proposed HTNE based on the Hawkes process and embedding lookup scheme, with the encoder defined in (53). Let \( H_t = ((v^{(1)}_1, t^{(1)}), (v^{(2)}_2, t^{(2)}), \ldots) \) be the sequence of
historical neighbors of node $v_i$ describing the formation of local topology centered at $v_i$, where
$$((v_i, v_{ij}^{(s)}), t^{(s)}) \in E_T \text{ and } t^{(s)} \leq t^{(s+1)}.$$ For simplicity, we denote the sequence of historical neighbors of $v_i$ before time step $t$ as $H_t(t)$. For each edge $((v_i, v_j), t) \in E_T$, HTNE defines the conditional intensity $\lambda_{ij}(t)$ as
$$\lambda_{ij}(t) \equiv \mu_{ij} + \sum_{(v_p, t_p) \in H_t(t)} \alpha_{pj} \kappa(t - t_p),$$
where $\mu_{ij} \equiv -||Z_{i,i} - Z_{j,j}||^2$ is the base intensity; $\kappa(t - t_p) = \exp(-\delta(t - t_p))$ is the kernel function with $\delta(t)$ as a learnable parameter w.r.t. node $v_i$; $\alpha_{pj}$ is a weight adjusting the contribution of each historical neighbor $(v_p, t_p)$, which is determined by an attention unit applied to $Z$ (see [54] for its details). HTNE is then optimized by maximizing the likelihood (i.e., minimizing the negative log-likelihood) w.r.t. historical topology $\{H_t | v_i \in V\}$ via the following loss:
$$\min_Z \mathcal{L}(G_T(t-L, t); Z) \equiv - \sum_{v_i \in V} \sum_{(v_j, t) \in H_t} \ln P(v_j | v_i, H_t(t); Z),$$
with $P(v_j | v_i, H_t(t); Z) \equiv \frac{\exp(\lambda_{ij}(t))}{\sum_{v_p \in V} \exp(\lambda_{ip}(t))}$, (57)
where the likelihood $P(v_j | v_i, H_t(t); Z)$ w.r.t. each edge $((v_i, v_j), t)$ is formulated by the softmax of $\lambda_{ij}(t)$. Negative sampling [80] is also used to derive an approximated version of the aforementioned loss similar to that of DySAT in (52). HTNE is a task-independent DNE approach that needs a user-defined decoder for TLP. As recommended in [54], one can use the weighted-L1 norm strategy in Table 1 to derive edge embedding $\{e_{ij}\}$ and train a downstream logistic regression classifier on $\{e_{ij}\}$ to build the decoder.

(2) **M2DNE.** Lu et al. [55] proposed M2DNE by extending HTNE to explore the micro- and macro-dynamics that describe the (i) formation of graph topology and (ii) evolution of graph scale in terms of the number of edges. The encoder of M2DNE is already defined in (53) following the embedding lookup scheme.

To optimize the embedding lookup table $Z$, M2DNE first formulates the micro-dynamics of topology using a strategy similar to that of HTNE. For each edge $((v_i, v_j), t) \in E_T$, it defines the conditional intensity $\lambda_{ij}(t)$ as
$$\lambda_{ij}(t) \equiv \mu_{ij} + \beta_{ij} \left[ \sum_{(v_p, t_p) \in H_t} \alpha_{pj}(t) \mu_{pj}(t - t_p) \right] + (1 - \beta_{ij}) \left[ \sum_{(v_q, t_q) \in H_t} \alpha_{qj}(t) \mu_{qj}(t - t_q) \right],$$
where $\mu_{ij}$, $H_t(t)$, and $\kappa(t - t_p)$ are with the same definitions as those of HTNE in (56); $\alpha_{pj}(t)$ and $\beta_{ij}$ are weights determined by two attention modules applied to $Z$ (see [55] for their details). Let $e(t)$ be the number of edges at time step $t$ and $\Delta e(t_i) \equiv e(t_{i+1}) - e(t_i)$ with $t_i$ and $t_{i+1}$ as two successive time steps. For simplicity, let $\Omega$ be the set of time steps w.r.t. the training set. The loss function of M2DNE can be described as
$$\min_Z \mathcal{L}(G_\Omega; Z) \equiv - \sum_{t \in \Omega} \sum_{((v_i, v_j), t) \in E_\Omega} \ln P(v_j | v_i, H_t(t), H_j(t); Z) \theta \sum_{t \in \Omega} (\Delta e(t) - \Delta \hat{e}(t))^2,$$
where $\Delta \hat{e}(t)$ is the number of new edges at time step $t$ and the predicted value $\Delta \hat{e}(t)$ derived from embedding $Z$ (see [55] for its details). Negative sampling [80] is also applied to approximate the first term
with a form similar to that of DySAT in (52). Based on the aforementioned settings, M2DNE is a task-independent DNE approach and has the same decoder as HTNE.

3.5.3 Summary of ESD-Based OTI Methods. Although CDTNE, HTNE, and M2DNE can support the fine-grained representations of dynamic topology via UESD, they still follow the embedding lookup scheme of task-independent DNE and OTI paradigm, where the embedding lookup table \( Z \) is the model parameter optimized for a fixed period and one prediction operation. When it comes to a new time step, we need to optimize \( Z \) from scratch to support TLP on the latest topology, which is inefficient for applications with real-time constraints. The embedding lookup scheme, where the dimensionality of \( Z \) is related to the number of nodes \( N \), indicates that these methods can only support the TLP in level-1 but cannot tackle the deletion and addition of nodes in level-2. As the decoders of most task-independent DNE techniques (see Table 1) merely consider the TLP on unweighted graphs, the aforementioned methods also fail to support the prediction of weighted topology.

3.6 UESD-Based OTOG Methods

Most UESD-based OTOG approaches utilize the inductive nature of some DL structures to qualify the embedding of each node as a function of time. In this subsection, we denote the embedding of node \( v_i \) at time step \( t \) as \( z_i(t) \in \mathbb{R}^{1 \times d} \). This type of method can be further categorized based on the DL technique used to capture the evolving patterns of UESD-based topology, which include the time-encoded sequential models and deep continuous-time processes.

3.6.1 Time-Encoded Sequential Models. Some related methods use DL structures designed for discrete sequential data (e.g., RNN and attention) to handle UESD-based topology, where the continuous time difference between edges is preserved via specific temporal encoding.

(1) TGAT. Xu et al. [29] proposed TGAT that combines attention with temporal encoding based on Bochner’s theorem [81] in harmonic analysis. It defines a translation-invariant kernel \( \langle \Phi(t_1), \Phi(t_2) \rangle \) using a continuous functional mapping \( \Phi(t) \equiv [\cos(\omega_1 t), \sin(\omega_1 t), \ldots, \cos(\omega_d t), \sin(\omega_d t)]/\sqrt{d} \) with \( \omega_1, \ldots, \omega_d \sim \text{i.i.d } p(\omega) \) (i.e., independently sampled via a common distribution). \( \Phi(\Delta t) \) is then used to encode the information of continuous time difference \( \Delta t \). TGAT also assumes that node attributes are available and fixed for all the time steps, which can be described by an attribute matrix \( X \) with the \( i \)-th row denoting the attributes of node \( v_i \).

TGAT follows a multi-layer structure. For each node \( v_i \) at time step \( t \), let \( h_i^{(k)}(t) \) and \( z_i^{(k)}(t) \) be the input and output of the \( k \)-th layer. \( H_i(t) \) denotes the historical neighbors of \( v_i \) before \( t \), with the same definition as that in (56). For \( v_i \) and its neighbor \((v_j,t_j) \in H_i(t)\), we let \( z_i^{(k-1)}(t) = [h_j^{(k-1)}(t)||\Phi(0)] \) and \( \hat{z}_j^{(k-1)}(t) = [h_j^{(k-1)}(t)||\Phi(t-t_j)] \) by concatenating the node embedding and corresponding time difference encoding \( \Phi(\Delta t) \). Given a dynamic graph \( G_t \) with fixed attributes \( X \), the encoder w.r.t. the \( k \)-th TGAT layer to obtain embedding \( z_i^{(k)} \) can be formulated as

\[
    z_i^{(k)}(t) = \text{Enc}^{(k)}(G_t) \equiv \text{MLP}\left(\left[h^{(k-1)}(t)||X_{i,:}\right]\right), \quad \text{with } h^{(k-1)}(t) \equiv \sum_{(v_j,t_j) \in H_i(t)} a_{ij} \hat{z}_j^{(k-1)}(t)W_v
\]

\[
    \text{and } a_{ij} \equiv \exp\left(\left(\hat{z}_j^{(k-1)}(t)W_q\right)^T\left(\hat{z}_j^{(k-1)}(t)W_k\right)\right) / \sum_{v_i \in H_i(t)} \exp\left(\left(\hat{z}_j^{(k-1)}(t)W_q\right)^T\left(\hat{z}_j^{(k-1)}(t)W_k\right)\right),
\]

where \( W_q, W_k, \) and \( W_v \) are learnable parameters for the linear mapping of query, key, and value in the attention unit. Concretely, the \( k \)-th layer first derives an intermediate representation \( h^{(k-1)}(t) \) via the attentive combination of embedding \( \hat{z}_j^{(k-1)} \) w.r.t. each historical neighbor \((v_j,t_j) \in H_i(t)\),
from which the temporal local topology of \( v_i \) can be preserved. \( h^{(k-1)}(t) \) is then concatenated with the attributes \( X_i \) of target node \( v_i \) and further fed into an MLP.

We denote the embedding given by the last layer as \( [z_i(t)] \). Let \( \Omega \) denote the set of time steps w.r.t. the training set. \( \text{TGAT} \) is a task-independent DNE method that can be trained by maximizing the likelihood of historical topology \( G_{\Omega} \). As recommended in [29], we can apply the following approximated loss function with negative sampling for the model optimization of \( \text{TGAT} \):

\[
\min_\delta \mathcal{L}(G_{\Omega}; \delta) \equiv \sum_{((v_i, v_j), t) \in E_{\Omega}} - \ln \left( \sigma \left( -z_i(t)z_j^T(t) \right) \right) - n_s \mathbb{E}_{v_p \sim \mathcal{P}_n} \left[ \ln (\sigma(z_i(t)z_p^T(t))) \right],
\]

where \( n_s \) is the number of negative samples; \( \mathcal{P}_n \) is the distribution of negative sampling; \( \sigma(\cdot) \) is the sigmoid function. One of the strategies described in Table 1 can be used to build the decoder of \( \text{TGAT} \). By utilizing the availability of node attributes and the inductive nature of attention, \( \text{TGAT} \) is able to tackle the TLP in level-2 with non-fixed node sets.

(2) \text{CAW}. Wang et al. [43] extended the TRW (used by CTDNE as described in Section 3.5.1) to a set-based anonymization version, which removes node identities to support inductive learning while preserving the local topology structures. \( \text{CAW} \) was proposed to handle the anonymized features of TRWs using RNNs. Different from that of CTDNE as illustrated in Figure 8(a), \( \text{CAW} \) adopts the reverse TRW such that \( \omega = (v_0(t), v_1(t), \ldots, v_K(t)) \) with \((v^{(r-1)}, v^{(r)}) \in E_f \) and \( (v^{(r-1)}) \geq (v^{(r)}) \). Namely, each TRW should be sampled in descending order of the time steps. For instance, in Figure 8(b), \((v_8, v_3, v_2, v_1)\) and \((v_8, v_2, v_1, v_3)\) are valid TRWs for \( \text{CAW} \).

Let \( S_{v_i} \) be the set of TRWs starting from node \( v_i \). For each edge \((v_i, v_j), t \in E_f \), consider a node \( v_p \in S_{v_i} \cup S_{v_j} \). \( \text{CAW} \) defines a set-based anonymized feature \( I(v_p; S_{v_i}, S_{v_j}) \equiv [g(v_p, S_{v_i}), g(v_p, S_{v_j})] \), where \( g(v_p, S_{v_i}) \equiv [c_0^p, S_{v_i}, \ldots, c_K^p, S_{v_i}] \), \( c_r^p, S_{v_i} \) denotes the frequency that \( v_p \) appears at the \( r \)-th position among all the TRWs in \( S_{v_i} \). For the example in Figure 8(b), there are 3 TRWs in \( S_{v_8} \). For node \( v_2 \in S_{v_8} \), one can obtain \( g(v_2, S_{v_8}) = [0, 2, 1, 0] \) because the frequencies that \( v_2 \) appears at positions \( [0, 1, 2, 3] \) in \( S_{v_8} \) are \( [0, 2, 1, 0] \). Accordingly, the set-based anonymization of a TRW \( \omega = (v_0(t), v_1(t), \ldots) \) is defined as \( \hat{\omega} \equiv (I(v_0(t); S_{v_0(t)}, S_{v_0(t)}), I(v_1(t); S_{v_1(t)}, S_{v_1(t)}), \ldots) \). For a node pair \((v_i, v_j)\), let \( \hat{W}_{ij} \) be the set of anonymized TRWs \( \hat{\omega} \) w.r.t. \( S_{v_i} \cup S_{v_j} \). Given \( \hat{W}_{ij} \), the encoder of \( \text{CAW} \) is defined as

\[
e_{ij}(t^{(1)}) = \text{Enc}^\prime (\hat{W}_{ij}) \equiv \frac{1}{|\hat{W}_{ij}|} \sum_{\hat{\omega} \in \hat{W}_{ij}} \text{VanRNN}([f(\omega(t))]|\Phi(t^{(r-1)} - t^{(r+2)}))_{r=0,\ldots,K-2} ,
\]

\[
f(\omega(t)) \equiv \text{MLP}(g(\omega(t), S_{\omega(t)})) + \text{MLP}(g(\omega(t), S_{\omega(t)})) \text{ w.r.t. } I(\omega(t); S_{\omega(t)}, S_{\omega(t)}),
\]

\[
\Phi(\Delta t) \equiv [\cos(\omega_1 \Delta t), \sin(\omega_2 \Delta t), \ldots, \cos(\omega_d \Delta t), \sin(\omega_d \Delta t)] ,
\]

where \( e_{ij}(t) \) is the auxiliary edge embedding of node pair \((v_i, v_j)\) at time step \( t \); \( f(\omega(t)) \) represents a function to process \( I(\omega(t); S_{\omega(t)}, S_{\omega(t)}) \) (i.e., the anonymized features of the \( r \)-th node in a TRW), which uses two MLPs with shared parameters to encode the effects of both source and destination nodes of an associated edge \((v^{(r)}, v^{(r+1)})\); \( \Phi(\Delta t) \) is the temporal encoding that preserves the information of continuous time difference \( \Delta t \) with the same definition as that of \( \text{TGAT} \); \( \text{VanRNN}() \) denotes the vanilla RNN (see supplementary materials for its details) used to handle features \([f(\omega(t))]|\Phi(t^{(r)} - t^{(r+1)}))_{r=0,\ldots,K} \text{ w.r.t. discrete TRWs} \).

One can apply an MLP to \( e_{ij}(t) \) to derive the estimated probability \( \hat{E}_{ij}^{t+r} \equiv P((v_i, v_j), t)|G_f \) that an edge \((v_i, v_j)\) appears at a future time step \((t + r)\) with \( 0 < r \leq \Delta \). Namely, the decoder of \( \text{CAW} \) is defined as

\[
\hat{E}_{ij}^{t+r} = \text{Dec}(e_{ij}(t + r)) \equiv \text{MLP}(e_{ij}(t + r)).
\]

Let \( \Omega \) be a sampled training set with both positive and negative samples in terms of edges \{((v_i, v_j), t)\}. \( \text{CAW} \) is a task-dependent method optimized by the following cross-entropy loss

ACM Computing Surveys, Vol. 56, No. 4, Article 89. Publication date: November 2023.
regarding TLP:

$$\min_{\delta} \mathcal{L}(\Omega; \delta) \equiv \frac{1}{|\Omega|} \sum_{((v_i, v_j), t) \in \Omega} -E_{ij}^t \ln \tilde{E}_{ij}^t + \left(1 - E_{ij}^{tr}\right) \ln \left(1 - \tilde{E}_{ij}^t\right),$$

(64)

where $E_{ij}^t \in \{0, 1\}$ is the corresponding ground-truth. Since the set-based anonymized features $\{I(v_p; S_{v_i}, S_{v_j})\}$ are shared by all the possible dynamic topology, CAW can support the TLP in level-2, handling the variation of node sets.

3.6.2 Deep Continuous-Time Processes. In addition to using deep sequential models to handle the UESD-based topology, other methods directly formulate continuous-time processes (e.g., Hawkes process described in (55) and ordinary differential equation) using DL structures.

1) DyRep. Trivedi et al. [42] adopted a DL model to formulate TPP and introduced DyRep. For edge $e = ((v_i, v_j), t)$, let $t$ denote the time step of another edge observed just before $t$. Similarly, let $t_i$ be the time step that node $v_i$ is observed just before $t$. When $v_i$ is first added (i.e., not previously observed), we let $t_i = 0$ and set the initial embedding $z_i(t_i)$ to be its node attributes (if available) or a random vector. $H_i(t)$ represents the set of historical neighbors of $v_i$ before $t$, with the same definition as that of HTNE in (56). For each edge $((v_i, v_j), t) \in E_t$, the encoder of DyRep derives the embedding $(z_i(t), z_j(t))$ w.r.t. the two induced nodes $(v_i, v_j)$ via

$$z_i(t) = Enc(G_t) = \sigma(h_i(t)W_1 + z_i(t_i)W_s + (t - t_i)W_q)$$

$$\text{with } h_i(t) = \max \{\sigma(a_{ij}^t \cdot MLP(z_j(t_j))) \forall (v_j, t_j) \in H_i(t)\}.$$  

(65)

The first term $h_i(t)W_l$ explores the local second-order proximity of $v_i$, where $h_i(t)$ is the attentive aggregation of embedding $(z_i(t_i))$ w.r.t. historical neighbors $H_i(t)$; $a_{ij}^t$ is the weight determined by an attention unit (see [42] for its details) to adjust the contribution of each historical neighbor $(v_i, t_i) \in H_i(t)$. The second term $z_i(t_i)W_s$ encodes the self-propagation that $z_i(t)$ evolves w.r.t. its previous position $z_i(t_i)$. The third term $(t - t_i)W_q$ ensures that $z_i(t)$ is updated smoothly during interval $(t - t_i)$. $\{W_l, W_s, W_q\}$ are learnable parameters. $\sigma(\cdot)$ is the sigmoid function.

In DyRep, $k = 0$ and $k = 1$ are used to denote the cases that (i) a new edge $(v_i, v_j)$ is first added and (ii) an old edge $(v_i, v_j)$ is observed again. To formulate the TPP using embedding $(z_i(t))$, DyRep defines the conditional intensity for each edge $((v_i, v_j), t)$ with type $k$ as

$$\lambda_{ij}^k(t) = f_k(\cdot) \{z_i(t) W_k\},$$

where $f_k(x) = \psi_k \ln(1 + \exp(x/\psi_k))$ is a non-linear function ensuring that the intensity $\lambda_{ij}^k(t)$ is positive; $\{\psi_k, W_k\}$ are learnable parameters associated with $k \in \{0, 1\}$. Let $\Omega$ be the collection of time steps associated with the training set. DyRep maximizes the likelihood (i.e., minimizing the negative log-likelihood) w.r.t. edges in $\Omega$ via the following loss:

$$\min_{\delta} \mathcal{L}(G_{\Omega}; \delta) \equiv -\sum_{e=((v_i, v_j), t) \in \Omega} \ln \left(\lambda_{ij}^k(e)(t)\right) + \int_0^t \left(\sum_{v_i \in V_{\Omega}} \sum_{v_j \in V_{\Omega}} \sum_{k \in \{0, 1\}} \lambda_{ij}^k(s)\right) ds,$$

(66)

where $\pi(e) \in \{0, 1\}$ maps $e$ to its type; the second term is the survival probability for events that do not happen. The decoder estimates the probability $\tilde{E}_{ij}^{t+r} \equiv P((v_i, v_j), \tau + r|G_{\Omega})$ that an edge $(v_i, v_j)$ appears at a time step $(\tau + r)$ via

$$\tilde{E}_{ij}^{t+r} \propto \text{Dec}([z_i(\tau + r)]) = \lambda_{ij}^k(\tau + r) \cdot \exp \left\{\int_{\tau}^{\tau+r} \lambda_{ij}^k(s)ds\right\}.$$  

(67)

As DyRep initializes the embedding of each newly added node by setting $z_i(\bar{t}_i = 0)$, it can handle the TLP in level-2, deriving prediction results for new unseent nodes.

2) TREND. In [56], TREND was proposed to formulate Hawkes process using a multi-layer GNN. Let $z_i^{(k-1)}(t)$ and $z_i^{(k)}(t)$ be the input and output of the $k$th layer for node $v_i$ at time step $t$. As
node attributes are assumed to be available and fixed for all time steps, we set \( z_i^{(0)}(t) = X_{i,:} \) \( H_i(t) \) and \( \kappa(t - t_j) \) denote the (i) sequence of historical neighbors and (ii) decaying kernel with the same definitions as those in \([56]\). The \( k \)th encoder layer can be described as
\[
\tilde{z}_i^{(k)}(t) = \text{Enc}^{(k)}(G_i, X) \equiv \sigma \left( \tilde{z}_i^{(k-1)}(t) W_s^{(k-1)} + \sum_{(v_j, t_j) \in H_i(t)} \tilde{z}_j^{(k-1)}(t_j) W_h^{(k-1)} \kappa(t - t_j) \right),
\]
where \( \kappa(t - t_j) = \kappa(t - t_j)/\sum_{(v_k, t_k) \in H_i(t)} \kappa(t - t_k) \) is the time decaying factor normalized over historical neighbors \( H_i(t) \); \([W_s, W_h] \) are learnable parameters; \( \sigma(\cdot) \) is the sigmoid function. The first and second terms are used for receiving the self-information and aggregating the historical neighbors of \( v_i \) before \( t \). Based on the embedding \([z_i(t)] \) given by the last encoder layer, the conditional intensity of Hawkes process w.r.t. each edge \((v_i, v_j), t \in E_i \) is formulated as
\[
\lambda_{ij}(t) \equiv \text{MLP}([z_i(t) - z_j(t)]^2; \theta(v_i, v_j, t)), \quad \text{with} \quad \theta(v_i, v_j, t) = [\alpha(v_i, v_j, t) + 1] \odot \theta_e + \beta(v_i, v_j, t),
\]
\[
\alpha(v_i, v_j, t) \equiv \text{MLP}([z_i(t)] [z_j(t)]), \quad \text{and} \quad \beta(v_i, v_j, t) \equiv \text{MLP}([z_i(t)] [z_j(t)]),
\]
where \( \odot \) denotes the element-wise square; \( \theta(v_i, v_j, t) \) is the set of model parameters of \( \lambda_{ij}(t) \); \( \theta_e \) is a learnable event prior; \( \alpha(v_i, v_j, t) \) and \( \beta(v_i, v_j, t) \) are defined as the scaling and shifting operators, which are MLPs with \([z_i(t)] [z_j(t)] \) as inputs, following the feature-wise linear modulation \([82]\) in meta-learning. In this setting, the model parameters \( \theta(v_i, v_j, t) \) of \( \lambda_{ij}(t) \) are set to be automatically adjusted according to the information encoded in \([z_i(t)] [z_j(t)] \).

Let \( \Omega \) be the set of time steps associated with the training set. \( \text{TREND} \) is optimized via the following loss integrating both (i) \text{event dynamics} (i.e., formation of edges) and (ii) \text{node dynamics} (i.e., growth of edges) w.r.t. each node:
\[
\min_{\Omega} \mathcal{L}(G_{ij}, \delta) \equiv \sum_{((v_i, v_j), t) \in \Omega} \left[ \mathcal{L}_e(v_i, v_j, t) + \theta_1(\mathcal{L}_n(v_i, t) + \mathcal{L}_n(v_j, t)) + \theta_2 \left( \|\alpha(v_i, v_j, t)\|^2_{l_2} + \|\beta(v_i, v_j, t)\|^2_{l_2} \right) \right],
\]
with \( \mathcal{L}_e(v_i, v_j, t) = -\ln(\lambda_{ij}(t)) - n_{s} \mathbb{E}_{v_k \sim \mathcal{P}_n}[\ln(1 - \kappa(t))] \), and \( \mathcal{L}_n(v_i, t) \equiv [e_t(t) - \text{MLP}(z_i(t))]^2 \).

\( \mathcal{L}_e(v_i, v_j, t) \) is the loss to capture \text{event dynamics} via a strategy similar to \([61]\), where \( n_s \) and \( \mathcal{P}_n \) also have the same definitions as those in \([61]\). \( \mathcal{L}_n(v_i, t) \) is the loss incorporating \text{node dynamics} that minimizes the error between (i) the number of links \( e_t(t) \) formed by \( v_i \) at time step \( t \) and (ii) the predicted value \( \hat{e}_i(t) \equiv \text{MLP}(z_i(t)) \) given by an MLP, with motivations similar to the optimization of \text{M2DNE} in \([59]\). Moreover, the \( l_2 \)-regularization is applied to \( \alpha(v_i, v_j, t) \) and \( \beta(v_i, v_j, t) \) to avoid over-fitting. \( \{\theta_1, \theta_2\} \) are tunable hyper-parameters.

The aforementioned definitions of encoder and loss function indicate that \( \text{TREND} \) is a \text{task-independent} DNE approach. As recommended in \([56]\), one can use the same decoder as \text{HTNE} to derive prediction results. Furthermore, \( \text{TREND} \) can also tackle the TLP in level-2 because it utilizes the inductive nature of GNNs and the availability of node attributes.

(3) \textbf{GSNOP. Neural ordinary differential equation (NODE)} \([83]\) is another continuous-time model that can be used to handle the USED-based topology. Let \( u(t) \) a variable at time step \( t \). NODE formulates the derivative of \( u(t) \) w.r.t. \( t \) using a DL module denoted as \( f_{\text{ODE}}(u(t), t) \equiv du(t)/dt \). Given a start time step \( t_0 \) and initial value \( u(t_0) \), NODE can derive \( u(t) \) at any time step \( t > t_0 \) by solving the following equation
\[
u(t) = \text{NODE}(f_{\text{ODE}}, u(t_0), t_0) \equiv u(t_0) + \int_{t_0}^{t} f_{\text{ODE}}(u(s), s) ds.
\]

One can obtain the approximated value of \( u(t) \) by applying a numeric solver (e.g., Euler or Runge-Kutta solver \([84]\)) denoted as \( u(t) \approx \text{ODESolver}(f_{\text{ODE}}, u(t_0), t_0, t) \). Luo et al. \([57]\) combined
NODE with existing DNE models and proposed GSNOP, which is a task-dependent DNE approach for TLP.

Let $g_{\text{DNE}}(\cdot)$ be the encoder of an existing DNE-based method (e.g., DySAT and TGAT reviewed in this survey) that can derive the embedding $z_i(t)$ for each node $v_i$ at time step $t$. The encoder of GSNOP generates auxiliary edge embedding $\tilde{e}_{ij}(t)$ for each node pair $(v_i, v_j)$ at a previous time step $t \leq \tau$ based on $(z_i, z_j(t))$, which can be described as

$$e_{ij}(t) = \text{Enc}(G_t, g_{\text{DNE}}, v_i, v_j, t) \equiv \text{MLP}([z_i(t)||z_j(t)||y_{ij}']) + \Phi(t),$$

with $z_i(t) \equiv g_{\text{DNE}}(G_t, v_i, t)$, where $y_{ij}' = 1$ if $((v_i, v_j), t) \in E_t$ and $y_{ij}' = 0$ otherwise (i.e., an auxiliary variable denoting the existence of an edge); $\Phi(t) \in \mathbb{R}^d$ is the temporal encoding with the same definition as that of TGAT. For a future time step $(\tau + r)$, the encoder further generates $e_{ij}(\tau + r)$ based on NODE with $e_{ij}(\tau)$ as the initial state, which can be formulated as

$$e_{ij}(\tau + r) = \text{Enc}(f_{\text{ODE}}, e_{ij}(\tau), \tau + r) \equiv e_{ij}(\tau) + \int_{\tau}^{\tau + r} f_{\text{ODE}}(e_{ij}(t), \tau) dt,$$

with $f_{\text{ODE}}(e_{ij}(t), \tau) \equiv \text{MLP}(e_{ij}(t) + \Phi(t))$. (73)

The decoder of GSNOP takes $(z_i(\tau), z_j(\tau))$ and $e_{ij}(\tau + r)$ as inputs and estimates the probability $\mathcal{E}_{ij}^{\tau + r} \equiv P(((v_i, v_j), \tau + r)|G_t)$ that an edge $(v_i, v_j)$ appears at a future time step $(\tau + r)$ via

$$\hat{\mathcal{E}}_{ij}^{\tau + r} = \text{Dec}(z_i(\tau), z_j(\tau), e_{ij}(\tau + r)) \equiv \text{MLP}([\hat{z}_i(\tau + r)||\hat{z}_j(\tau + r)]),$$

with $\hat{z}_i(\tau + r) \equiv \text{MLP}([z_i(\tau)||\hat{e}_{ij}(\tau + r)]), \hat{e}_{ij}(\tau + r) \sim \mathcal{N}(\mu_{ij}^{\tau + r}, \sigma_{ij}^{\tau + r}), \mu_{ij}^{\tau + r} \equiv \text{MLP}(e_{ij}(\tau + r))$ and $\sigma_{ij}^{\tau + r} \equiv 0.1 + 0.9 \text{MLP}(e_{ij}(\tau + r))$. (74)

Following a structure similar to the variational autoencoder (VAE) [85], the decoder first constructs a Gaussian distribution $\mathcal{N}(\mu_{ij}^{\tau + r}, \sigma_{ij}^{\tau + r})$, with the mean $\mu_{ij}^{\tau + r}$ and variance $\sigma_{ij}^{\tau + r}$ formulated by $e_{ij}(\tau + r)$. Another embedding $\hat{e}_{ij}(\tau + r)$ is then sampled from $\mathcal{N}(\mu_{ij}^{\tau + r}, \sigma_{ij}^{\tau + r})$ and concatenated with $(z_i(\tau), z_j(\tau))$ to derive $\hat{\mathcal{E}}_{ij}^{\tau + r}$ using MLPs. Let $\Omega$ be the set of time steps w.r.t. the training set. Similar to VAE, the loss function of GSNOP maximizes the evidence lower bound (ELBO) [85] (i.e., minimizing the negative ELBO) to optimize model parameters $\delta$, which can be described as

$$\min_{\delta} \mathcal{L}(G_{\delta}; \Omega) \equiv -\mathbb{E}_{Q(\hat{e}_{ij}^e | e_{ij}^g)} \left[ \ln \hat{\mathcal{E}}_{ij}^{\tau + r} \right] + \sum_{ij} \text{KL} \left[ Q(\hat{e}_{ij}^e | e_{ij}^g) \| P(e_{ij}^g) \right],$$

with $Q(\hat{e}_{ij}^e | e_{ij}^g) \sim \mathcal{N}(\mu_{ij}^{\tau + r}, \sigma_{ij}^{\tau + r})$ and $P(e_{ij}^g) \approx e_{ij}(\tau + r)$. (75)

The first term maximizes the expectation of $\ln \hat{\mathcal{E}}_{ij}^{\tau + r}$, which is equivalent to minimizing the cross-entropy between $\hat{\mathcal{E}}_{ij}^{\tau + r}$ and ground-truth, with a form similar to (64). The second term minimizes the KL-divergence between $Q(e_{ij}^e | e_{ij}^g)$ and $P(e_{ij}^g)$, where $Q(e_{ij}^e | e_{ij}^g)$ is the posterior distribution of $e_{ij}(\tau + r)$ given previous embedding $e_{ij}(t)$ with $t \leq \tau$, which is estimated by $\mathcal{N}(\mu_{ij}^{\tau + r}, \sigma_{ij}^{\tau + r})$; $P(e_{ij}^g)$ is the prior distribution of $e_{ij}(\tau + r)$ estimated via the NODE in (73).

3.6.3 Summary of UESD-Based OTOG Methods. In summary, the aforementioned approaches (e.g., TGAT, CAV, DyRep, TREND, and GSNOP) can support the fine-grained representation of dynamic topology to handle the rapid topology variation via UESD. As they adopt the OTOG paradigm based on the inductive nature of DL structures and attributes/features shared by all the possible nodes, they can tackle the TLP in level-2 with non-fixed node sets while having the potential to satisfy the real-time constraints of systems. However, since they still rely on stochastic processes (e.g., TPP and TRW) on unweighted graphs, they cannot explore the weighted topology and support the advanced TLP on weighted graphs. Due to limitations of the OTOG paradigm, this kind of method may also have the risk of failing to capture the latest evolving patterns of graphs.
4 ADVANCED TOPICS & FUTURE DIRECTIONS

In this section, we first summarize some advanced topics in recent research based on our review of existing TLP methods in Section 3. Furthermore, several possible future directions are highlighted at the end of this section.

4.1 Advanced Research Topics

4.1.1 Prediction of Weighted Topology. Most existing TLP methods merely focus on the prediction of unweighted topology. Some of them are inapplicable to the TLP on weighted graphs. On the one hand, the encoders and loss functions of some methods cannot capture the variation of weighted topology. For instance, most UESD-based approaches rely on stochastic processes defined on unweighted graphs (e.g., TRW and TPP introduced in Sections 3.5.1 and 3.5.2), which do not have the hypotheses regarding the evolution of edge weights. On the other hand, the decoders of some approaches (e.g., ctRBM, DyRep, and TGAT) are only designed for unweighted graphs, treating TLP as the binary edge classification, which can only derive the probability that an edge will appear in a time step but cannot predict the corresponding edge weight.

Although some ESSD-based methods (e.g., GrNMF, TMF, dyngraph2vec, and DDNE) can still support the TLP on weighted graphs by using adjacency matrices \(A_t\) to describe the weighted topology, they can only derive low-quality prediction results. Most of them are optimized via error minimization objectives that minimize the reconstruction error between the training ground-truth \(A_{t+1}\) and prediction result \(\tilde{A}_{t+1}\) as illustrated in (22) and (34). Qin et al. [44] argued that these objectives cannot tackle the following wide-value-range and sparsity issues.

**Wide-Value-Range Issue.** In weighted graphs, edge weights may have a wide value range (e.g., \([0, 2000]\)). There may also be a non-ignorable portion of edges with small weights. However, error minimization objectives are only sensitive to large edge weights but fail to distinguish the scale difference between small weights. For instance, the scale difference between \((1, 2)\) is larger than that between \((1990, 2000)\), although the latter case has a larger error. From the view of pre-allocating system resources, failing to distinguish the scale difference of edge weights may have the risks of (i) allocating much more resources than the real demand of a link or (ii) not allocating enough resources for a link.

**Sparsity Issue.** In an adjacency matrix \(A_t\), small and zero elements have different physical meanings. \((A_t)_{ij} = 0\) indicates that there is no edge between \((v^t_i, v^t_j)\). A small element \((A_t)_{ij} > 0\) implies that there is still an edge between \((v^t_i, v^t_j)\) but the edge weight is small. The topology of some real-world systems may be sparse with a non-ignorable portion of zeros in \(A_t\). Since error minimization objectives are only sensitive to large weights, methods that are only optimized via these objectives may also fail to distinguish the difference between zero and small weights in \(A_t\). For resource pre-allocation, failing to distinguish between zero and small weights may also have the risks of (i) allocating resources for non-existent links or (ii) not allocating resources for existing links.

Due to the aforementioned issues, most existing methods (e.g., GrNMF, TMF, dyngraph2vec, and DDNE) can only generate low-quality prediction results \(\{A_{t+1}\}\) (in terms of adjacency matrices) that fail to distinguish between small and zero weights for weighted graphs, and thus can only support the coarse-grained resource allocation.

To derive high-quality prediction results for weighted snapshots, some advanced methods (e.g., GCN-GAN and IDEA) combine error minimization objectives with adversarial learning. Recent progress in high-resolution video prediction [72] has demonstrated that GAN can help minimize the scale difference between pixel values in images via its adversarial process and thus can derive high-resolution predicted video frames. Intuitively, it is also expected that **adversarial learning can help distinguish between the scale difference of weights in adjacency matrices**.

ACM Computing Surveys, Vol. 56, No. 4, Article 89. Publication date: November 2023.
Figure 9 illustrates the weighted TLP on the UCSB-MeshNet dataset from a campus wireless mesh network (see supplementary materials for its details), where we visualize adjacency matrices of (i) historical snapshots, (ii) prediction results of DDNE, GCN-GAN, and IDEA, as well as (iii) ground-truth of one example prediction operation. To highlight the difference between small and zero weights, we set all 0s to $-500$ in each visualized adjacency matrix, where dark blue, light blue, and yellow denote zero, small, and large weights. Similar to DDNE, error minimization is also a classic objective of video prediction, which can only derive low-quality prediction results (e.g., the blurred image and dense adjacency matrix of DDNE in Figure 9). In contrast, the integration of GAN can effectively help derive high-quality results close to ground-truth (e.g., the clear image and sparse adjacency matrices of GCN-GAN and IDEA in Figure 9).

However, existing methods that can support weighted TLP still rely on ESSD using adjacency matrices to describe weighted topology, which can only achieve coarse-grained representations of dynamic graphs compared with UESD.

### 4.1.2 Dealing with the Variation of Node Sets

Some TLP methods (e.g., ctRBM, GrNMF, DDNE, and CTDNE) assume that the node set $V$ of a dynamic graph is known and fixed for all time steps. They rely on some techniques (e.g., matrix factorization and embedding lookup scheme introduced in Sections 3.3 and 3.5), in which the dimensionality of model parameters is related to the number of nodes. As the dimensionality of model parameters should be fixed for all the time steps, these approaches can only support the TLP in level-1, failing to tackle the variation of node sets.

However, most real-world systems allow the addition and deletion of entities. To some extent, methods with ESSD can tackle the variation of node sets using "large" adjacency matrices $\{A_t\}$ induced by the union of all the associated nodes, where there may be isolated nodes without edges in some snapshots. Such a naive strategy may have unnecessary high space complexity. Moreover, it can only derive the prediction results w.r.t. the previously observed nodes but cannot be generalized to new unseen nodes in future time steps.

Inductive inference of dynamic graphs is an advanced topic in recent research. Typical inductive TLP methods (e.g., DyRep, TGAT, and IDEA) use the available node attributes and the inductive nature of DL structures (e.g., GNNs and attention), in which the dimensionality of model parameters is not related to the number of nodes, to ensure a trained model can be directly generalized to new unseen nodes. Therefore, these inductive approaches can support the TLP in level-2, dealing with the variation of node sets. Nevertheless, most of them still rely on the availability of node attributes but lack discussions regarding how to handle the case without attributes. Only a few methods (e.g., CAW) can extract features that are shared by all the possible nodes from the raw dynamic topology to support inductive inference.
4.2 Future Research Directions

**Simultaneous Prediction of Node and Edge Sets.** As defined in Section 2.2, node sets w.r.t. future links are usually assumed to be given in existing research. For instance, $V^\tau_{r^{+\Delta}}$ and $V_{r^{(r^{+\Delta})}}$ are inputs of the TLP in level-2 as described in (3) and (4). TLP aims at predicting possible future edges induced by the given node sets. In addition to edge sets, the dynamics of a graph may also include the variation of node sets. In some real applications, the set of system entities in future time steps may also be unknown. For the TLP in level-2, simultaneously predicting the sets of future nodes and edges is more challenging and seldom studied in recent research. Moreover, the quality evaluation criteria of existing research are still based on the given future node sets (e.g., $V^\tau_{r^{+\Delta}}$ or $V_{r^{(r^{+\Delta})}}$). New quality metrics are also required to evaluate the prediction of both node and edge sets.

**Combining with Other Dynamic Inference Tasks.** In addition to TLP, there are some other inference tasks on dynamic graphs (e.g., dynamic community detection [86–88], anomaly detection [89, 90], and traffic prediction [91, 92]). In particular, the traffic prediction task is usually formulated as the dynamic node attribute prediction, while some TLP methods (e.g., EvolveGCN, DySAT, IDEA, and TGAT) assume that node attributes are available and only focus on the prediction of dynamic topology. Simultaneously optimizing a unified model to tackle multiple inference tasks (e.g., combining TLP with dynamic attribute prediction) is a promising future direction. It is usually expected that the optimization of multiple associated tasks can further improve the performance of each other [93].

For some settings considering both graph topology and attributes (e.g., prediction of future topology and attributes), recent studies [94, 95] have revealed complicated correlations between the two sources. On the one hand, attributes may carry complementary information beyond topology to support the better performance of a task. On the other hand, it may also result in unexpected performance decline due to the inconsistent noises hidden in attributes. Hence, it is promising to develop an advanced model that can adaptively adjust the contributions of different heterogeneous information sources (e.g., dynamic topology and attributes) according to the possible ‘mismatch’ between them.

**Adaptive Selection of Sampling Rate and Execution Frequency.** As introduced in Section 2.1, we need to select a fixed time interval (or corresponding sampling rate) between successive snapshots when using ESSD to abstract a real-world system as a dynamic graph. Accordingly, the execution frequency of TLP can then be determined, where we execute one prediction operation once it comes to a new time step. Usually, the time interval (or sampling rate) is set according to the minimum duration of interactions in the system, which may result in high space complexities with many redundant descriptions of dynamic topology. In contrast, when using UESD, we sample a corresponding edge once there is a new interaction in the system without specifying the sampling rate, but still need to set a proper execution frequency of TLP. However, most related studies did not consider the selection of sampling rate and execution frequency, assuming that they are already determined by concrete applications or datasets. Only a few research considered the setting of sampling rate for ESSD (e.g., based on the autocorrelation between snapshots [96]).

It is promising to explore an adaptive strategy for the selection of sampling rate and execution frequency according to the evolution and overhead of a system. For instance, we can set a high sampling rate and execution frequency in peak hours of a system to accurately capture its evolving patterns. When the system is not busy, we can set a relatively low sampling rate and execution frequency to minimize the system overhead.

**New Learning Paradigms.** As introduced in Section 2.4, OTI and OTOG are widely-used learning paradigms of the existing TLP techniques. On the one hand, OTI methods can effectively capture the latest variation of topology but may fail to satisfy the real-time constraints of...
applications due to their high complexities of online training. On the other hand, OTOG approaches have the potential to satisfy the real-time constraints in their online generalization without additional model optimization, but may also fail to capture the latest characteristics as the topology evolves. It is promising to develop new learning paradigms that integrate the advantages of both OTI and OTOG methods, where recent progress in online learning [97, 98] and continual learning [99, 100] can be applied.

**Theoretical Analysis of Prediction Quality.** As discussed in Section 4.1, TLP can be used to pre-allocate key resources for better system performance. In particular, some methods (e.g., GCN-GAN and IDEA) focus on how to derive high-quality weighted links to support fine-grained resource allocation. However, some real-world systems may also have the reliability requirements [101, 102] while a TLP model still has the risk of making wrong predictions. Therefore, theoretical analysis on the bound of prediction accuracy and error can help determine whether a TLP method can satisfy reliability requirements of a system.

5 CONCLUSION

In this survey, we comprehensively reviewed existing representative TLP methods. We first gave the formal definitions regarding (i) data models of dynamic graphs (i.e., ESSD and UESD), (ii) task settings of TLP (i.e., level-1 and -2), and (iii) learning paradigms of related research (i.e., DI, OTI, and OTOG). Based on these definitions, we further introduced a fine-grained hierarchical taxonomy that categorizes existing methods in terms of (i) data models, (ii) learning paradigms, and (iii) techniques, covering multiple aspects. From a generic perspective, a unified encode-decoder framework was proposed to formulate all the methods reviewed in this survey. Each method can be described by an encoder, a decoder, and a loss function, where different methods only differ in terms of these components. Based on this unified framework, we also refactored or implemented some TLP approaches and served the community with an open-source project OpenTLP. Finally, we summarized some advanced topics in recent research and future research directions. Due to space limit, we elaborate on (i) additional preliminaries (e.g., quality evaluation and classic techniques), (ii) applications, and (iii) public datasets of TLP in supplementary materials.

REFERENCES

[1] Srijan Kumar, Francesca Spezzano, V. S. Subrahmanian, and Christos Faloutsos. 2016. Edge weight prediction in weighted signed networks. In Proceedings of the 16th IEEE International Conference on Data Mining. IEEE, 221–230.

[2] Srijan Kumar, Bryan Hooi, Disha Makhija, Mohit Kumar, Christos Faloutsos, and VS Subrahmanian. 2018. Rev2: Fraudulent user prediction in rating platforms. In Proceedings of the 11th ACM International Conference on Web Search and Data Mining. 333–341.

[3] Pierre Borgnat, Guillaume Dewaele, Kensuke Fukuda, Patrice Abry, and Kenjiro Cho. 2009. Seven years and one day: Sketching the evolution of internet traffic. In Proceedings of the 2019 IEEE INFOCOM Conference on Computer Communications. IEEE, 711–719.

[4] Arunan Sivanathan, Hassan Habibi Gharakheili, Franco Loi, Adam Radford, Chamith Wijenayake, Arun Vishwanath, and Vijay Sivaraman. 2018. Classifying IoT devices in smart environments using network traffic characteristics. IEEE Transactions on Mobile Computing 18, 8 (2018), 1745–1759.

[5] Petter Holme and Jari Saramäki. 2012. Temporal networks. Physics Reports 519, 3 (2012), 97–125.

[6] Kai Lei, Meng Qin, Bo Bai, and Gong Zhang. 2018. Adaptive multiple non-negative matrix factorization for temporal link prediction in dynamic networks. In Proceedings of the ACM SIGCOMM 2018 Workshop on Network Meets AI and ML. 28–34.

[7] Kai Lei, Meng Qin, Bo Bai, Gong Zhang, and Min Yang. 2019. GCN-GAN: A non-linear temporal link prediction model for weighted dynamic networks. In Proceedings of the 2019 IEEE INFOCOM Conference on Computer Communications. IEEE, 388–396.

[8] Mattia G. Campana and Franca Delmastro. 2017. Recommender Systems for Online and Mobile Social Networks: A survey. Online Social Networks and Media 3–4, (2017), 75–97. DOI: https://doi.org/https://doi.org/10.1016/j.osnem.2017.10.005
[9] Jianling Wang, Kaize Ding, Liangjie Hong, Huan Liu, and James Caverlee. 2020. Next-item recommendation with sequential hypergraphs. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval. 1101–1110.

[10] Isaiah J. King and H. Howie Huang. 2023. Euler: Detecting network lateral movement via scalable temporal link prediction. IEEE Transactions on Privacy and Security 26, 3 (June 2023). DOI: https://doi.org/10.1145/3588771

[11] Weifeng Gao, Zhiwei Zhao, Zhengxin Yu, Geyong Min, Minghang Yang, and Wenjie Huang. 2020. Edge-computing-based channel allocation for demand-driven IoT networks. IEEE Transactions on Industrial Informatics 16, 10 (2020), 6693–6702.

[12] Connor Vinchoff, Nathan Chung, Tyler Gordon, Liam Lyford, and Michal Aibin. 2020. Traffic prediction in optical networks using graph convolutional generative adversarial networks. In Proceedings of the 22nd International Conference on Transparent Optical Networks. IEEE, 1–4.

[13] Michal Aibin, Nathan Chung, Tyler Gordon, Liam Lyford, and Connor Vinchoff. 2021. On short-and long-term traffic prediction in optical networks using machine learning. In Proceedings of the 2021 International Conference on Optical Network Design and Modeling. IEEE, 1–6.

[14] Michael Hunter Ashby and Jenna A. Bilbrey. 2021. Geometric Learning of the Conformational Dynamics of Molecules using Dynamic Graph Neural Networks.

[15] Victor Martinez, Fernando Berzal, and Juan-Carlos Cubero. 2016. A survey of link prediction in complex networks. ACM Computing Surveys 49, 4 (2016), 1–33.

[16] Ajay Kumar, Shashank Sheshar Singh, Kuldeep Singh, and Bhaskar Biswas. 2020. Link prediction techniques, applications, and performance: A survey. Physica A: Statistical Mechanics and its Applications 553, (2020), 124289. DOI: https://doi.org/10.1016/j.physa.2020.124289

[17] Peng Cui, Xiao Wang, Jian Pei, and Wenwu Zhu. 2018. A survey on network embedding. IEEE Transactions on Knowledge and Data Engineering 31, 5 (2018), 833–852.

[18] Xiaoyi Li, Nan Du, Hui Li, Kang Li, Jing Gao, and Aidong Zhang. 2014. A deep learning approach to link prediction in dynamic networks. In Proceedings of the 2014 SIAM International Conference on Data Mining. SIAM, 289–297.

[19] Giang Hoang Nguyen, John Boaz Lee, Ryan A. Rossi, Nesreen K. Ahmed, Eunyee Koh, and Sungchul Kim. 2018. Continuous-time dynamic network embeddings. In Companion Proceedings of the 2018 Web Conference. 969–976.

[20] Seyed Mehran Kazemi, Rishab Goel, Kshitij Jain, Ivan Kobyzev, Akshay Sethi, Peter Forsyth, and Pascal Poupart. 2020. Representation learning for dynamic graphs: A survey. Journal of Machine Learning Research 21, 70 (2020), 1–73.

[21] Guotong Xue, Ming Zhong, Jianxin Li, Jia Chen, Chengshuai Zhai, and Ruochen Kong. 2022. Dynamic network embedding survey. Neurocomputing 472, (2022), 212–223. DOI: https://doi.org/10.1016/j.neucom.2021.03.138

[22] Claudio D. T. Barros, Matheus R. F. Mendonça, Alex B. Vieira, and Artur Ziviani. 2021. A survey on embedding dynamic graphs. ACM Computing Surveys 55, 1 (2021), 1–37.

[23] Joakim Skarding, Bogdan Gabrys, and Katarzyna Musial. 2021. Foundations and modeling of dynamic networks using dynamic graph neural networks: A survey. IEEE Access 9 (2021), 79143–79168.

[24] Shawndra Hill, Deepak K Agarwal, Robert Bell, and Chris Volinsky. 2006. Building an effective representation for applications, and performance: A survey. Journal of Computational and Graphical Statistics 15, 3 (2006), 584–608.

[25] Umang Sharan and Jennifer Neville. 2008. Temporal-relational classifiers for prediction in evolving domains. In Proceedings of the 8th IEEE International Conference on Data Mining. IEEE, 540–549.

[26] Taisong Li, Jiawei Zhang, Philip S. Yu, Yan Zhang, and Yonghong Yan. 2018. Deep dynamic network embedding for link prediction. IEEE Access 6, (2018), 29219–29230. DOI: https://doi.org/10.1109/ACCESS.2018.2839770

[27] Palash Goyal, Sujit Rokka Chhetri, and Arquimedes Canedo. 2020. dyngraph2vec: Capturing network dynamics using dynamic graph representation learning. Knowledge-Based Systems 187, (2021), 104816. DOI: https://doi.org/10.1016/j.knossys.2019.06.024

[28] Shengjie Min, Zhan Gao, Jing Peng, Liang Wang, Ke Qin, and Bo Fang. 2021. STGSN — A Spatial-Temporal graph neural network framework for time-evolving social networks. Knowledge-Based Systems 214 (2021), 106746. DOI: https://doi.org/10.1016/j.knossys.2021.106746

[29] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achian. 2020. Inductive representation learning on temporal graphs. In Proceedings of the 8th International Conference on Learning Representations.

[30] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, and Hao Yang. 2020. DySat: Deep neural representation learning on dynamic graphs via self-attention networks. In Proceedings of the 13th International Conference on Web Search and Data Mining. 519–527.

[31] Solaymani Divakaran and Anuraj Mohan. 2020. Temporal link prediction: A survey. New Generation Computing 38, 1 (2020), 213–258.
[33] Andrea Rossi, Denilson Barbosa, Donatella Firmani, Antonio Matinata, and Paolo Meritaldo. 2021. Knowledge graph embedding for link prediction: A comparative analysis. ACM Transactions on Knowledge Discovery from Data 15, 2 (2021), 1–49.

[34] Borui Cai, Yong Xiang, Longxiang Gao, He Zhang, Yunfeng Li, and Jianxin Li. 2023. Temporal knowledge graph completion: A survey. In Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI). ijcai.org, 6545–6553. DOI: https://doi.org/10.24963/ijcai.2023/734

[35] Zhengyu Huang, Aimin Zhou, and Guixu Zhang. 2012. Non-negative matrix factorization: A short survey on methods and applications. In Proceedings of the International Symposium on Intelligence Computation and Applications. Springer, 331–340.

[36] Pierre De Handschutter, Nicolas Gillis, and Xavier Siebert. 2021. A survey on deep matrix factorizations. Computer Science Review 42 (2021), 100423. DOI: https://doi.org/https://doi.org/10.1016/j.cosrev.2021.100423

[37] Jonatan A. González, Francisco J. Rodríguez-Cortés, Otmar Cronie, and Jorge Mateu. 2016. Spatio-temporal point process statistics: A review. Spatial Statistics 18 (2016), 505–544. DOI: https://doi.org/https://doi.org/10.1016/j.spasta.2016.10.002

[38] Baichuan Yuan, Hao Li, Andrea L. Bertozzi, P Jeffrey Brantingham, and Mason A. Porter. 2019. Multivariate spatiotemporal Hawkes processes and network reconstruction. SIAM Journal on Mathematics of Data Science 1, 2 (2019), 356–382.

[39] Liming Zhang, Liang Zhao, Shan Qin, Dieter Pfoser, and Chen Ling. 2021. TG-GAN: Continuous-time temporal graph deep generative models with time-validity constraints. In Proceedings of the 2021 Web Conference. 2104–2116.

[40] Wenchao Yu, Wei Cheng, Charu C. Aggarwal, Kai Zhang, Haifeng Chen, and Wei Wang. 2018. Netwalk: A flexible deep embedding approach for anomaly detection in dynamic networks. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2672–2681.

[41] Yao Ma, Ziyi Guo, Zhaocong Ren, Jiliang Tang, and Dawei Yin. 2020. Streaming graph neural networks. In Proceedings of the 43rd ACM International SIGIR Conference on Research and Development in Information Retrieval. 719–728.

[42] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. 2019. DyRep: Learning representations over dynamic graphs. In Proceedings of the 7th International Conference on Learning Representations.

[43] Yanbang Wang, Yen-Yu Chang, Yunyu Liu, Jure Leskovec, and Pan Li. 2021. Inductive representation learning in temporal networks via causal anonymous walks. In Proceedings of the 9th International Conference on Learning Representations.

[44] Meng Qin, Chaorui Zhang, Bo Bai, Gong Zhang, and Dit-Yan Yeung. 2023. High-quality temporal link prediction for weighted dynamic graphs via inductive embedding aggregation. IEEE Transactions on Knowledge and Data Engineering 35, 9 (2023), 9378–9393. DOI: https://doi.org/10.1109/TKDE.2023.3238360

[45] David Liben-Nowell and Jon Kleinberg. 2003. The link prediction problem for social networks. In Proceedings of the 12th ACM International Conference on Information and Knowledge Management. 556–559.

[46] Sheng Gao, Ludovic Denoyer, and Patrick Gallinari. 2011. Temporal link prediction by integrating content and structure information. In Proceedings of the 20th ACM International Conference on Information and Knowledge Management. 1169–1174.

[47] Xiaoke Ma, Shiyin Tan, Xianghua Xie, Xiaoxiong Zhong, and Jingjing Deng. 2022. Joint multi-label learning and feature extraction for temporal link prediction. Pattern Recognition 121 (2022), 108216. DOI: https://doi.org/https://doi.org/10.1016/j.patcog.2021.108216

[48] Xiaoke Ma, Penggang Sun, and Yu Wang. 2018. Graph regularized nonnegative matrix factorization for temporal link prediction in dynamic networks. Physica A: Statistical Mechanics and its Applications 496 (2018), 121–136. DOI: https://doi.org/https://doi.org/10.1016/j.physa.2017.12.092

[49] Nahla Mohamed Ahmed, Ling Chen, Yulong Wang, Bin Li, Yun Li, and Wei Liu. 2018. DeepEye: Link prediction in dynamic networks based on non-negative matrix factorization. Big Data Mining and Analytics 1, 1 (2018), 19–33.

[50] Wenchao Yu, Charu C. Aggarwal, and Wei Wang. 2017. Temporally factorized network modeling for evolutionary network analysis. In Proceedings of the 10th ACM International Conference on Web Search and Data Mining. 455–464.

[51] Wenchao Yu, Wei Cheng, Charu C. Aggarwal, Haifeng Chen, and Wei Wang. 2017. Link prediction with spatial and temporal consistency in dynamic networks. In Proceedings of the 26th International Joint Conference on Artificial Intelligence. 3343–3349.

[52] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanazashi, Tim Kaler, Tao Schardl, and Charles Leiserson. 2020. Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In Proceedings of the 34th AAAI Conference on Artificial Intelligence. 5363–5370.
[54] Yuan Zuo, Guannan Liu, Hao Lin, Jia Guo, Xiaqian Hu, and Junjie Wu. 2018. Embedding temporal network via neighborhood formation. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2857–2866.

[55] Yuanfu Lu, Xiao Wang, Chuan Shi, Philip S. Yu, and Yanfang Ye. 2019. Temporal network embedding with micro-and macro-dynamics. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management. 469–478.

[56] Zhihao Wen and Yuan Fang. 2022. TRENDS: Temporal event and node dynamics for graph representation learning. In Proceedings of the ACM Web Conference 2022. 1159–1169.

[57] Linhao Luo, Gholamreza Haffari, and Shirui Pan. 2023. Graph sequential neural ode process for link prediction on dynamic and sparse graphs. In Proceedings of the 6th ACM International Conference on Web Search and Data Mining. 778–786.

[58] Leo Katz. 1953. A new status index derived from sociometric analysis. Psychometrika 18, 1 (1953), 39–43.

[59] Daniel D. Lee and H. Sebastian Seung. 1999. Learning the parts of objects by non-negative matrix factorization. Nature 401, 6755 (1999), 788–791.

[60] Daniel D. Lee and H. Sebastian Seung. 2000. Algorithms for Non-negative matrix factorization. In Proceedings of the Advances in Neural Information Processing Systems 13, MIT Press, 556–562. Retrieved from https://proceedings.neurips.cc/paper/2000/hash/9d1152547c0bde01830b7e8b6d0024c-Abstract.html

[61] Deng Cai, Xiaofei He, Jiawei Han, and Thomas S. Huang. 2010. Graph regularized nonnegative matrix factorization for data representation. IEEE Transactions on Pattern Analysis and Machine Intelligence 33, 8 (2010), 1548–1560.

[62] Wei Cheng, Kai Zhang, Haifeng Chen, Guofei Jiang, Zhenghang Chen, and Wei Wang. 2016. Ranking causal anomalies via temporal and dynamical analysis on vanishing correlations. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 805–814.

[63] Nan Zhang, Shifei Ding, Jian Zhang, and Yu Xue. 2018. An overview on restricted Boltzmann Machines. Neurocomputing 275 (2018), 1186–1199. DOI: https://doi.org/10.1016/j.neucom.2017.09.065

[64] Benyamin Ghojogh, Ali Ghodsi, Fakhrri Karray, and Mark Crowley. 2022. Restricted Boltzmann Machine and Deep Belief Network: Tutorial and Survey.

[65] Felix A. Gers, Jürgen Schmidhuber, and Fred Cummins. 2000. Learning to forget: Continual prediction with LSTM. Neural Computation 12, 10 (2000), 2451–2471.

[66] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling.

[67] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, andIllia Polosukhin. 2017. Attention is All you Need. In Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, 5998–6008. Retrieved from https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

[68] Zhaoyang Niu, Guoqiang Zhong, and Hui Yu. 2021. A review on the attention mechanism of deep learning. Neural Computing 452 (2021), 48–62. DOI: https://doi.org/10.1016/j.neucom.2021.03.091

[69] Geoffrey E. Hinton. 2002. Training products of experts by minimizing contrastive divergence. Neural Computation 14, 8 (2002), 1771–1800.

[70] Junying Chen, Jian Zhang, Xuanheng Xu, Chenbo Fu, Dan Zhang, Qingpeng Zhang, and Qi Xuan. 2019. E-LSTM-d: A deep learning framework for dynamic network link prediction. IEEE Transactions on Systems, Man, and Cybernetics: Systems 51, 6 (2019), 3699–3712.

[71] Thomas N. Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. In Proceedings of the 4th International Conference on Learning Representations.

[72] Michael Mathieu, Camille Couprie, and Yann LeCun. 2015. Deep multi-scale video prediction beyond mean square error. In Proceedings of the 3rd International Conference on Learning Representations.

[73] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8–13 2014, Montreal, Quebec, Canada, 2672–2680. Retrieved from https://proceedings.neurips.cc/paper/2014/hash/5ca3e9b1226f1f8f0f6494e97b1afccf3-Abstract.html

[74] Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein generative adversarial networks. In Proceedings of the 34th International Conference on Machine Learning, PMLR, 214–223.

[75] Petar Velickovic, Guillem Cucurull, Aranatz Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In Proceedings of the 6th International Conference on Learning Representations (ICLR), OpenReview.net. Retrieved from https://openreview.net/forum?id=rJXMPikCZ

[76] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems.
[100] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. 2021. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 7 (2021), 3366–3385.

[101] Marvin Rausand and Arnljot Hoyland. 2003. *System Reliability Theory: Models, Statistical Methods, and Applications*. Vol. 396. John Wiley & Sons.

[102] Waqar Ahmad, Osman Hasan, Usman Pervez, and Junaid Qadir. 2017. Reliability modeling and analysis of communication networks. *Journal of Network and Computer Applications* 78, (2017), 191–215. DOI: https://doi.org/https://doi.org/10.1016/j.jnca.2016.11.008

Received 17 October 2022; revised 24 August 2023; accepted 18 September 2023