Towards Machine Learning on Temporal Graphs

Shenyang Huang, School of Computer Science

McGill University, Montreal

11, 2024

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of

Doctor of Philosophy

Abstract

Temporal graphs model the complex interactions observed in real-world networks such as social networks, transaction platforms, and traffic networks. Temporal graph learning methods are important for many applications such as recommending products to users, detecting fraudulent transactions in a financial network and predicting future events in a knowledge base.

One important aspect of temporal graph is the evolution of the graph structure over time. This thesis introduces state-of-the-art methods for the *change point detection* task which detects timestamps where the structure of a temporal graph deviates significantly from the norm. In Chapter 4, a novel spectral learning method based on the graph Laplacian matrix is proposed to capture structural changes over time. Then, in Chapter 5, we tackle the scalability challenge of this task by leveraging efficient approximations of the spectral properties of the Laplacian and successfully scaling to networks with millions of edges. Finally, in Chapter 6, we design the first change point detection method for multi-view dynamic graphs by merging spectral information from all views via the power mean aggregation. We demonstrate that all of our approaches identify significant real world events which impacts the graph evolution on political networks, traffic networks, citation networks and other complex networks.

Another central topic of the thesis examines the limitations of existing evaluation of temporal graph learning methods, including a lack of large and diverse datasets and simplistic negative samples. To address these challenges, in Chapter 7, we propose the Temporal Graph Benchmark (TGB), a collection of challenging and diverse benchmark

datasets for realistic, reproducible, and robust evaluation for machine learning on temporal graphs. TGB provides an automated pipeline and challenging evaluation for existing and future methods. Shortly after its release, TGB became a widely used benchmark in temporal graph learning while receiving constant improvements based on community feedback. To include more types of temporal graph into TGB, we also extend the benchmark to include temporal knowledge graph and temporal heterogeneous graph datasets in Chapter 8, addressing the lack of datasets and benchmarks in these communities respectively. Lastly, in Chapter 9, we further compare different temporal graph methods by proposing the first framework which unifies snapshot-based and event-based methods, enabling direct comparison between the two distinct lines of research. Through these projects, we provide a solid foundation for future research on temporal graphs, closely aligning the datasets, evaluation, and method designs toward real world applications.

Abrégé

Les graphes temporels modélisent les interactions complexes observées dans les réseaux du monde réel tels que les réseaux sociaux, les plateformes de transaction et les réseaux de circulation. Les méthodes d'apprentissage pour les graphes temporels sont importantes pour de nombreuses applications telles que la recommandation de produits aux utilisateurs, la détection de transactions frauduleuses dans un réseau financier et la prédiction d'événements futurs dans une base de connaissances.

Un aspect important des graphes temporels est l'évolution de la structure du graphe dans le temps. Cette thèse présente des méthodes de pointe pour la tâche de détection de rupture, qui permet de détecter les moments où la structure d'un graphe temporel s'écarte de manière significative de la norme. Le chapitre 4 propose une nouvelle méthode d'apprentissage spectral basée sur la matrice laplacienne du graphe pour saisir les changements structurels au fil du temps. Dans le chapitre 5, nous relevons le défi du passage à l'échelle de cette tâche en exploitant des approximations efficaces des propriétés spectrales du laplacien et nous parvenons à étendre notre méthode à des réseaux comportant des millions d'arêtes. Enfin, dans le chapitre 6, nous concevons la première méthode de détection de rupture pour les graphes dynamiques multi-vues en fusionnant les informations spectrales de toutes les vues via l'agrégation de la moyenne de puissance. Nous démontrons que toutes nos approches identifient des événements importants du monde réel qui ont eu un impact sur l'évolution des graphes des réseaux politiques, des réseaux de circulation, des réseaux de citations et d'autres réseaux complexes.

Un autre thème central de cette thèse examine les limites de l'évaluation actuelle des méthodes d'apprentissage pour graphes temporels, notamment le manque d'ensembles de données larges et diversifiés et les échantillons négatifs simplistes. Pour relever ces défis, nous proposons au chapitre 7 le *Temporal Graph Benchmark* (TGB), une collection de larges ensembles de données de référence difficiles et diversifiés pour une évaluation réaliste, reproductible et robuste de l'apprentissage automatique sur les graphes temporels. TGB fournit un pipeline automatisé et une évaluation dynamique des méthodes existantes et futures. Peu de temps après son lancement, TGB est devenu une référence largement utilisée dans le domaine de l'apprentissage des graphes temporels, tout en bénéficiant d'améliorations constantes basées sur le retour d'information de la communauté. Afin d'inclure davantage de types de graphes temporels dans TGB, nous étendons également le benchmark aux ensembles de données de graphes de connaissances temporelles et de graphes hétérogènes temporels au chapitre 8, afin de remédier au manque d'ensembles de données et de benchmarks dans ces communautés. Enfin, au chapitre 9, nous comparons les différentes méthodes de graphes temporels en proposant le premier cadre qui unifie les méthodes temporelles discrètes et continues, ce qui permet une comparaison directe entre ces deux lignes de recherche distinctes. Grâce à ces projets, nous fournissons une base solide pour les recherches futures sur les graphes temporels, en alignant étroitement les ensembles de données, l'évaluation et la conception des méthodes sur les applications du monde réel.

Acknowledgements

I would like to sincerely thank my Ph.D. supervisors, Reihaneh Rabbany and Guillaume Rabusseau for their unwavering support, continued encouragement, uncountable valuable advice and feedback throughout my journey as a Ph.D. student. Their constructive feedback, insightful discussions, generous support of my ideas and pursuits shaped my growth as a research and as an individual. In addition, I would like to thank my Ph.D. community for their valuable feedback: Jian Tang and Joelle Pineau.

I am also incredibly thankful to my partner, Yining Wang who always encourages and supports me in my personal and academic journey, helping me through tough times and enriching my life outside of research. I would also like to extend heartfelt appreciation to my parents Zhihong Shen and Huaqiang Huang, for making my academic journey possible and unwavering faith in my pursuits. In addition, I would also like to thank my grandparents who always cheer me on. I would also like to thank Yifu Zhang, Naiyun Zhang, Kuan-Chieh (Jackson) Wang, Julie Yoon, Akshay Gopalakrishnan and Valentin Lehericy for being excellent friends and companions along my research journey and personal life.

For my lab members in both of my supervisors' groups, I am very grateful to everyone for paper discussions, feedback and help along the way. I would like to especially thank Farimah Poursafaei for being a close collaborator and always brainstorm with me, Kellin Pelrine and Aarash Feizi for all the help on workshop organization and research discussions, Razieh Shirzadkhani, Jacob Danovitch, Charlotte Ding, Abby Leung and Sacha

Levy for exploring many interesting research projects. I am also very grateful to Tianyu Li for suggestions and help on how to write this thesis.

I am very fortunate to have collaborated with a diverse group of researchers during my PhD and I would like to express my gratitude for all of their advice and feedback. I am very grateful to Alireza Makhzani, Dominique Beaini and Matthias Fey for being great mentors to me during my internships at the Vector Institute, Valence Labs and kumo.ai. I am also very thankful to Weihua Hu, Emanuele Rossi, Mikhail Galkin, Julia Gastinger and Ali Parviz for your research feedback and making the TGB projects possible.

Last but not least, I would like to thank the Canadian Institute for Advanced Research (CIFAR AI chair program), Natural Sciences and Engineering Research Council of Canada (NSERC) Postgraduate ScholarshipDoctoral (PGS D) Award and Fonds de recherche du Québec - Nature et Technologies (FRQNT) Doctoral Award for supporting my research funding, conference travels and more. I would also like to thank the administrative and finance staff at Mila and McGill university, such as Ann Jack, for helping me and answering my logistics questions.

Contents

List of Figures

List of Tables

List of Abbreviations

GNN . . . . . Graph Neural Network

GCN . . . . . Graph Convolution Network [107]

RNN . . . . . Recurrent Neural Network

LSTM . . . . . Long Short Term Memory

TGL . . . . . Temporal Graph Learning

TKG . . . . . Temporal Knowledge Graph

THG . . . . . Temporal Heterogeneous Graph

CPD . . . . . Change Point Detection

LAD . . . . . Laplacian Anomaly Detection [93]

SCPD . . . . . Scalable Change Point Detection [92]

MultiLAD . . Multi-view Laplacian Anomaly Detection [91]

TGB . . . . . Temporal Graph Benchmark [63]

UTG . . . . . Unifying Temporal Graph [95]

MRR . . . . . Mean Reciprocal Rank

SVD . . . . . Singular Value Decomposition

Chapter 1

Introduction

“We are all now connected by the Internet, like neurons in a giant brain", as said by Stephen Hawking to describe our increasingly digitized, interconnected world. Large social platforms connect users across the world with millions of messages sent each minute [225, 165, 115], transaction networks receive thousands of purchase orders every second [162, 86] and large human contact networks enable us to predict future trends of ongoing pandemics [84, 26, 119]. New users or connections are constantly introduced to these evolving networks and can be modeled as temporal graphs (also referred to as dynamic graphs, used interchangeably in this thesis).

Temporal Graph Learning (TGL) has important applications in areas such as recommendation systems [162], fraud detection [222], epidemic modeling [26] and traffic forecasting [34]. In applied settings, a TGL method is often required to be *efficient*, *effective* and *scalable*. In addition to these criteria, there are also key challenges such as how to properly model the graph evolution and how to learn temporal dependencies. Significant advances in Machine Learning (ML) are often accelerated by the availability of standardized and large-scale benchmarks, exemplified by Open Graph Benchmark [88] in graph learning, ImageNet [40] in computer vision and GLUE benchmark [202] in Natural Language Processing. However, in temporal graph learning, commonly used datasets and benchmarks are lacking, resulting in over-optimistic performances [158].

This thesis starts by introducing notations and background for static graph learning and temporal graph learning in Chapter 2. Then, a literature overview of related work is presented in Chapter 3. In this thesis, our contributions can be broadly divided into two parts. In the first part, we aim to design methods that satisfy real-world criteria of being efficient, effective, and scalable. Then in the second part, we propose benchmarks and frameworks to evaluate TGL methods with the above criteria and provide a solid foundation for future TGL research.

In the first part, we focus on the task of *Change Point Detection* (CPD) on dynamic graphs. Specifically, the objective is to identify time points where the graph structure undergoes drastic changes. In Chapter 4, we first propose an efficient and effective solution to the CPD problem by leveraging the spectral properties of the graph. Then we tackle the scalability aspect in Chapter 5 via efficient approximations of the spectral density. Lastly, we extend our approach to the setting of multi-view dynamic graphs in Chapter 6 where multiple networks can describe the same underlying relations. Across political networks, traffic networks, social networks, and more, our proposed CPD methods detect time points corresponding to significant real-world events that impact the temporal graph structure.

In the next part, we address the limitations in current evaluation for temporal graph learning, namely *lack of datasets* and *simplistic evaluation*. To this end, we design novel benchmarks to include datasets from a wide range of domains and sizes and realistic evaluation protocols. In Chapter 7, we first propose the Temporal Graph Benchmark, designed for ML tasks on homogeneous temporal graphs. Then in Chapter 8, we introduce novel datasets and evaluation protocols for two additional types of temporal graphs: Temporal Knowledge Graphs (TKGs) and Temporal Heterogeneous Graphs (THGs). Lastly, in Chapter 9, we propose a framework to unify temporal graph methods designed for different data types and provide a first comprehensive comparison between these categories of methods.

In the current chapter, we start by stating our contribution to original knowledge. Then the contributions of authors for the publications and submissions presented here is discussed. Lastly, my contributions in other publications not included in this thesis are also presented.

Contribution to Original Knowledge. Here we summarize our contributions to temporal graph learning following a chapter-based structure.

- **LAD**. In Chapter 4, we introduce a novel CPD method, Laplacian Anomaly Detection (LAD). LAD computes the Singular Value Decomposition (SVD) of the graph Laplacian to obtain a low-dimensional graph representation. To the best of our knowledge, this is the first time that the Laplacian spectrum is used for the change point detection task. LAD explicitly captures both short-term and long-term temporal relations to model the abrupt and gradual changes in dynamic networks via two sliding windows of different sizes. We extensively evaluate LAD on two synthetic experiments and three real-world datasets. Empirically, LAD achieves better or competitive performance than state-of-the-art methods on all datasets.

- **SCPD**. In Chapter 5, we propose Scalable Change Point Detection (SCPD), a novel scalable CPD method that detects both structural and node attribute anomalies in dynamic graphs. SCPD utilizes the distribution of eigenvalues (also known as the spectral density) of the Laplacian matrix as a low dimensional embedding of each graph snapshot, efficiently approximated by the density of states framework. In this way, SCPD scales linearly with respect to the number of edges. Empirically, we show that SCPD achieves state-of-the-art performance for synthetic experiments and identifies a number of major wars from the co-authorship network of the History research community. Lastly, to the best of our knowledge, SCPD is the first method to incorporate node attributes into change point detection for dynamic graphs. On our original COVID-flight dataset, SCPD leverages the country code of airports (nodes) to identify traffic disturbances due to flight restrictions specific to countries such as China and US.

- **MultiLAD**. In Chapter 6, we extend LAD to the multi-view setting by proposing MultiLAD, the first CPD method for multi-view dynamic networks. MultiLAD aggregates the singular values of the normalized Laplacian matrices through scalar power mean and identifies the most informative singular values from each view. Empirical evaluation of MultiLAD shows that MultiLAD, 1). gains increased performance from additional views, 2). is highly robust to noise, and 3). significantly outperforms state-of-the-art single-view baselines and their multi-view extensions. We apply MultiLAD to two real-world multi-view traffic networks. We demonstrate that MultiLAD correctly detects major real-world traffic disruptions such as the implementation of stay-at-home orders for COVID-19 and Christmas Day.

- **TGB**. In Chapter 7, we present the Temporal Graph Benchmark (TGB), a collection of challenging and diverse benchmark datasets for realistic, reproducible, and robust evaluation for machine learning on temporal graphs. TGB includes datasets coming from a diverse range of domains and spanning both edge and node-level tasks. We contributed seven novel datasets which are orders of magnitude larger than previously existing ones in terms of the number of edges, nodes, and timestamps. We propose an improved and standardized evaluation pipeline motivated by real-world applications. For the dynamic link property prediction task, it is observed that model performances can vary drastically across datasets. On the node affinity prediction task, simple heuristics often outperform state-of-the-art TG methods, thus leaving ample room for the development of future methods targeting this task. This project is open-sourced and will continue to improve based on community feedback. We have constructed a public website, code repository and software package to directly download and process datasets.

- **TGB 2.0**. In Chapter 8, we present TGB 2.0, a novel benchmark designed for future link prediction on multi-relational temporal graphs. Building upon the foundations of TGB where only *single-relation temporal graphs* are included, TGB2 introduces *multi-*

relational temporal graph datasets. TGB2 adds four novel TKG datasets and four novel THG datasets spanning five domains and are significantly larger than existing ones. In addition, TGB2 includes an evaluation pipeline for multi-relational temporal graphs, which automates the dataset downloading, processing, and benchmarking process for seamless reproducibility. The main insight from our experiments is that for both THGs and TKGs, all methods (apart from heuristics) fail to scale to our largest datasets, highlighting the need for more research on scalable methods. Surprisingly, the heuristic baselines perform competitively with more sophisticated methods.

- **UTG**. In Chapter 9, we aim to bridge the gap between event-based and snapshot-based models by providing a unified framework to train and evaluate them to predict future events on any type of temporal graph. To achieve this, we propose *Unified Temporal Graph* (UTG), a framework that unifies snapshot-based and event-based temporal graph models under a single umbrella, enabling models developed for one representation to be applied effectively to datasets of the other. Moreover, we propose a novel UTG training strategy to boost the performance of snapshot-based models in the streaming setting. The comparison between event-based and snapshot-based models shows that event-based models with joint neighborhood structural features can achieve strong performance while snapshot-based models are more efficient in comparison. This opens up novel research directions to combine the benefits of these two approaches and design efficient and effective TG learning methods.

Contribution of Authors. Here we describe the contributions of authors in each publication or submitted work in this thesis.

- Chapters 1, 2, 3 are written for this thesis.

- Chapter 4 is based on the KDD 2020 conference paper [93] authored by me, Yasmeen Hitti, Guillaume Rabusseau and Reihaneh Rabbany. As the first author on this work, I developed the LAD method, conducted all experiments, and wrote the paper. Yasmeen

supported me in collecting data for the Canadian Parliament network data and edited the paper. Guillaume and Reihaneh provided feedback and guidance on the project.

- Chapter 5 is based on the PAKDD 2023 conference paper [92] authored by me, Jacob Danovitch, Guillaume Rabusseau and Reihaneh Rabbany. As the first author of this work, I developed the SCPD method, conducted all experiments and wrote the paper. Jacob supported me by helping with dataset collection. Guillaume and Reihaneh provided feedback and guidance on the project.

- Chapter 6 is based on the TKDD journal paper [91] authored by me, Samy Coulombe, Yasmeen Hitti, Reihaneh Rabbany and Guillaume Rabusseau. This is a paper extending LAD into the multi-view dynamic graph setting. As the first author, I developed the MultiLAD approach, ran experiments as well as drafted the paper. Samy helped me collect the NYC transit dataset and conducted experiments. Yasmeen helped with the data collection process from the Canadian Parliament. Guillaume and Reihaneh provided feedback and guidance on the project.

- Chapter 7 is based on the NeurIPS 2023 dataset and benchmark track paper [94] authored by me, Farimah Poursafaei, Jacob Danovitch, Matthias Fey, Weihua Hu, Emanuele Rossi, Jure Leskovec, Michael Bronstein, Guillaume Rabusseau and Reihaneh Rabbany. Being the co-first author on this work, I designed the benchmark, collected and processed datasets, ran experiments for baselines such as TGN, DyRep, and EdgeBank, made the TGB website as well as drafted the paper. As my co-first author, Farimah helped with designing the benchmark, ran experiments for all other baselines, and edited the paper. Jacob helped collected the `tgbl-review` as well as helped with dataset hosting, website hosting and making the code available as a Python pip package. Matthias, Weihua, and Emanuele were involved in idea brainstorming and task design. Jure, Michael, Guillaume and Reihaneh provided feedback for this work.

- Chapter 8 is based on our recent work to appear in the Dataset and Benchmark track of NeurIPS 2024 [63]. This work is authored by me, Julia Gastinger, Mikhail Galkin, Erfan

Loghmani, Ali Parviz, Farimah Poursafaei, Jacob Danovitch, Emanuele Rossi, Ioannis Koutis, Heiner Stuckenschmidt, Reihaneh Rabbany and Guillaume Rabusseau. I am the co-first author on this work, I have developed the improved evaluation pipeline for TKG and THG tasks, collected and processed datasets in this work, implemented the code for TGB and edited / revised the paper draft. Julia is my co-first author who implemented the baselines for TKG methods, provided advice on data collection, benchmarked TKG methods, helped with designing evaluation protocols as well as writing the paper draft. Mikhail was involved in providing guidance in the data collection process as well as evaluation design. Erfan, Ali and Farimah helped with running baselines for THG experiments. Jacob provided assistance in data hosting. Emanuele, Ioannis, Heiner, Reihaneh and Guillaume provided feedback for the project.

- Chapter 9 is based on the recent work that was submitted for review as a conference paper to Learning on Graph (LoG) 2024. This work is authored by me, Farimah Poursafaei, Reihaneh Rabbany, Guillaume Rabusseau and Emanuele Rossi. As the co-first author of this work, I have designed and implemented the UTG pipeline to convert any TG method to work on both CTDG or DTDG representations, drafted the paper, ran experiments of some baselines, designed the UTG training scheme to improve snapshot-based models. As my co-first author, Farimah helped me improve the implementation, ran baseline experiments and edited the paper. Reihaneh, Guillaume and Emanuele provided feedback for the project.

- Chapter 10 is written for this thesis.

Contributions in Other Publications. During my PhD, I also collaborated on several other projects which led to the following publications (not included in this thesis):

- Second author of “Incorporating dynamic flight network in SEIR to model mobility between populations", published in Applied Network Science, Special issue on Epidemics Dynamics & Control on Networks [42]. In this work, we propose to modify

the commonly-used SEIR disease model to account for the dynamic flight network, by estimating the imported cases based on the air traffic volume and the test positive rate. I contributed to paper writing, dataset collection as well as the design of the proposed Flight-SEIR method.

- Co-first author of “Towards Better Evaluation for Dynamic Link Prediction", published at NeurIPS 2022 Datasets and Benchmarks Track [158]. In this work, we examined the limitations of current evaluation pipelines for dynamic link prediction and proposed novel datasets, negative samples, heuristic baseline as well as visualizations to facilitate more robust evaluation. I contributed to the paper writing, idea conception and discussion, implementation and collection of novel datasets for the paper.

- Second author of “Temporal Graph Analysis with TGX", published in WSDM 2024 Demo Track [181]. In this project, we designed a Python package designed specifically for analyzing temporal graphs. TGX provides built-in datasets, data processing features such as discretization function and node subsampling as well as different temporal graph measures. As the second author, I contributed to project conception, code implementation, API design, paper writing. I am also the lead maintainer of this library.

- Second author of “Towards Foundational Models For Molecular Learning on Large-scale Multi-task Datasets", published in ICLR 2024 [7]. In this project, we present seven novel datasets pushing the boundaries in both the scale and the diversity of supervised labels for molecular learning, with nearly 100 million molecules and over 3000 sparsely defined tasks. We also present the Graphium library which simplifies the process of building and training molecular machine learning models for multi-task and multi-level molecular datasets. As the second author, I contributed to project brainstorming, paper writing and was a core contributor to Graphium.

- Third author of “GraphPulse: Topological representations for temporal graph property prediction", published in ICLR 2024 [176]. Here, we explore the task of predicting fu-

ture properties of a given temporal graph, by introducing a novel framework named GraphPulse. As the third author, I contributed to idea conception (including the temporal graph property prediction task), code implementation as well as paper writing.

Chapter 2

Background and Overview

In this chapter, we introduce the necessary background and notations for this thesis. We start by introducing static graph notations in Section 2.1. Then we provide the formulation of temporal graphs in Section 2.2. Lastly, we discuss related background in random graph models in Section 2.3.

2.1 Static Graph Notations

A (static) graph is often represented as a tuple of node and edge sets, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V} = \{v_1, v_2, \ldots, v_{|\mathcal{V}|}\}$ is the set of nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges (or links). Nodes can also have attributes (or features), denoted by $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times N_a}$ where N_a is the number of attributes. Optionally, edges might also contain attributes $\mathbf{M} \in \mathbb{R}^{|\mathcal{E}| \times k_m}$, where k_m is the size of the edge attribute vector. Graphs with node or edge attributes are often referred to as *attributed graphs*.

A common way of representing a graph is by using an *adjacency matrix* $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ indicating the edges of the graph where $\mathbf{A}[i][j] = 0$ if $(v_i, v_j) \notin \mathcal{E}$, otherwise $\mathbf{A}[i][j] = w \in \mathbb{R}^+$, representing the weight w of the edge. In *unweighted graphs*, all edge weights are equal to one. Otherwise, the graph is considered as *weighted graph*. The *degree* of a node

v_i is defined as the sum of the weights of edges it is connected to, $\sum_{j=1}^{|\mathcal{V}|} \mathbf{A}[i][j]$. The *degree matrix* $\mathbf{D} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ is a diagonal matrix where $\mathbf{D}[i][i]$ is the degree of node v_i.

For an *undirected graph*, the adjacency matrix $\mathbf{A}$ is symmetric, i.e., $\mathbf{A}[i][j] = \mathbf{A}[j][i]$ for all i and j. In a *directed graph*, each edge has a direction going from a source node to a destination node, thus its adjacency matrix is no longer symmetric. A *bipartite graph* is a graph where its vertex set $\mathcal{V}$ can be partitioned into exactly two disjoint sets $\mathcal{V}_1$ and $\mathcal{V}_2$ where all edges are only connecting nodes across $\mathcal{V}_1$ and $\mathcal{V}_2$, i.e., $\mathcal{E} \subset \mathcal{V}_1 \times \mathcal{V}_2$.

Another common matrix is the (unnormalized) Laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{A}$ where $\mathbf{D}$ is the degree matrix and $\mathbf{A}$ is the adjacency matrix of $\mathcal{G}$. The symmetric normalized Laplacian $\mathbf{L}_{sym}$ is defined as follows,

$$\mathbf{L}_{sym} = \mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$$

and guarantees the matrix is symmetric while its eigenvalues are bounded in the range $[0, 2]$. Note that for directed graphs, the Laplacian matrix $\mathbf{L}$ is defined differently [32].

2.2 Temporal Graph Background

Representations of temporal graphs are often divided into two categories: discrete time dynamic graphs (DTDGs) and continuous time dynamic graphs (CTDGs) [104]. DTDGs are represented by an ordered sequence of static graph snapshots (see Section 2.2.1) while CTDGs are represented as timestamped edge streams (see Section 2.2.2).

DTDGs are used in applications where the temporal graph structure is sampled at regular time intervals, i.e. daily, weekly, monthly and so on. Therefore DTDG methods specialize in processing a sequence of graph snapshots and capturing the changes between snapshots [223, 61, 153, 217]. In contrast, CTDGs are used to model real-world networks that continue to receive new data at any second of the day such as social networks, mobile networks and transaction networks. Therefore, CTDG methods aggregate

features from the temporal neighborhood (connected nodes with close time proximity) and constantly update the node embedding as newer edges arrive [165, 206, 133].

2.2.1 Discrete Time Dynamic Graphs

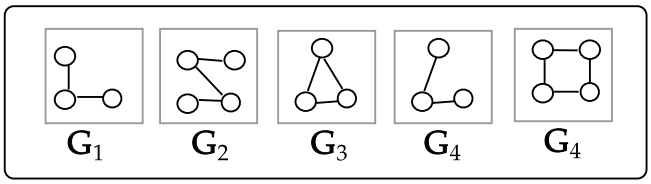


Figure 2.1: Illustration of a discrete time dynamic graph, represented as a sequence of graph snapshots.

A *discrete time dynamic graph* is often defined as a sequence of graph snapshots $\{\mathcal{G}_t\}_{t=1}^{T}$ (illustrated in Figure 2.1). Each $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t, \mathbf{X}_t)$ represents the graph at time $t \in [1 \dots T]$, where $\mathcal{V}_t$ and $\mathcal{E}_t$ are the set of nodes and edges respectively. The optional node attribute matrix is $\mathbf{X}_t \in \mathbb{R}^{|\mathcal{V}_t| \times N_a}$ where N_a is the number of attributes. An edge $(i, j, w) \in \mathcal{E}_t$ connects node i and node j at time t with weight w. We use $\mathbf{A}_t \in \mathbb{R}^{|\mathcal{V}_t| \times |\mathcal{V}_t|}$ to denote the adjacency matrix of $\mathcal{G}_t$. Note that in dynamic graphs, the active set of nodes and edges at each timestamp t might differ thus we use the subscript t to represent the graph $\mathcal{G}_t$. For simplicity, $\mathcal{G}_t$ implicitly refers to a snapshot in a discrete time dynamic graph.

Multi-view Dynamic Graphs. A *multi-view dynamic graph* $\mathcal{G} = \{\mathcal{G}_{t,r}\}_{t=1,r=1}^{T,m}$ is given by a collection of graph snapshots across timestamps and different views or relations. $\mathcal{G}_{t,r}$ is the graph snapshot from the view (or relation) r at time step t. At each time step, each snapshot $\mathcal{G}_{t,r} = (\mathcal{V}_{t,r}, \mathcal{E}_{t,r})$ consists of the set of nodes $\mathcal{V}_{t,r}$ and a set of edges $\mathcal{E}_{t,r} \subset \mathcal{V}_{t,r} \times \mathcal{V}_{t,r}$. Each edge $e = (i, j, w) \in \mathbf{E}_{t,r}$ is then defined as the connection between node i and node j at timestamp t in the view r with weight $w \in \mathbb{R}^+$. The adjacency matrix $\mathbf{A}_{t,r} \in \mathbb{R}^{|\mathcal{V}_{t,r}| \times |\mathcal{V}_{t,r}|}$ represents edges in $\mathcal{E}_{t,r}$. When the number of views $m = 1$, a single view dynamic graph is recovered which is the setting used for LAD and SCPD.

2.2.2 Continuous Time Dynamic Graphs

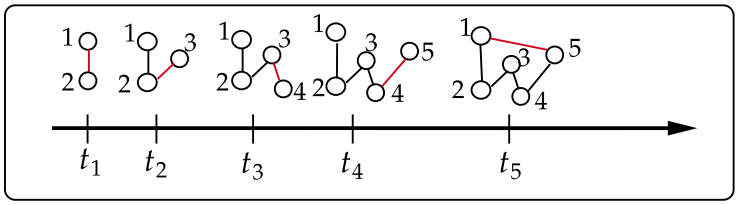


Figure 2.2: Illustration of a continuous time dynamic graph, represented as a stream of timestamped edges.

Continuous Time Dynamic Graphs (CTDGs) are represented as timestamped streams of edges, i.e., $\mathcal{G} = \{(s_0, d_0, t_0), (s_1, d_1, t_1), \ldots, (s_T, d_T, T)\}$, where each tuple consists of a source node s, destination node d and timestamp t, the timestamps are ordered ($0 \leq t_1 \leq t_2 \leq ... \leq T$) [158, 104] (illustrated in Figure 2.2). Note that the graph can be directed or undirected, and weighted or unweighted. $\mathcal{G}_t$ is the cumulative graph constructed from all edges in the stream until time t with nodes $\mathbf{V}_t$ and edges $\mathbf{E}_t$. Optionally, there can be node features $\mathbf{X}_t \in \mathbb{R}^{|\mathbf{V}_t| \times k_x}$, where k_x is the size of the node feature vector, and edge features $\mathbf{M}_t \in \mathbb{R}^{|\mathbf{E}_t| \times k_m}$, where k_m is the size of the edge feature vector. We consider a fixed chronological split to form the training, validation and test set.

2.3 Random Graph Models

Random graph models are powerful tools in the field of graph theory [57]. Natural networks often exhibit complex patterns and random graphs are used to model such behavior. In Chapter 4, 5 and 6, we use random graph models to generate temporal graphs for evaluation of the change point detection task. Here, we provide the background and formulations of these random graph models, starting with the earliest Erdős–Rényi (ER) model which was named after the authors [51].

Erdős-Rényi Model. In the Erdős-Rényi (ER) model, a graph is constructed by connecting nodes randomly, and each edge is connected independently of every other edge with a fixed probability [51]. The ER model is represented by $\mathbf{G}(n, p)$ with parameters p and n where p is the probability of any edge being present and n is the number of nodes in the graph. Equivalently, the probability of generating an ER graph with n nodes and m edges is:

$$p^m(1-p)^{\binom{n}{2}-m}.$$

The probability parameter p is in the range of $[0, 1]$ where the larger the p is, the more likely it is for an edge to exist between any pair of nodes in the graph.

Stochastic Block Model. The Stochastic Block Model (SBM) is a random graph generative model used to produce graphs containing community structures [83]. A community is defined as a subset of nodes within the graph that contains denser connections between nodes when compared to the rest of the network. The SBM is defined by the following parameters:

- The number of nodes n.
- A partition of the node set $\mathcal{V}$ into k disjoint subsets, each representing a community.
- a symmetric probability matrix $\mathbf{P} \in \mathbb{R}^{k\times k}$ representing the inter-community and intra-community connectivity for each block.

For simplicity, we consider the special case where the inter-community p_{in} and intra-community p_{ex} connectivity is the same for every community in the graph. Therefore, the probability matrix $\mathbf{P}$ can be described by p_{in} and p_{ex} only where p_{in} fills the diagonal entries and p_{ex} fills the off-diagonal ones. Common structural shifts in temporal graphs such as a change in the community structure or an increase in cross-community connections can be represented directly as changes to the SBM parameters. Therefore, we leverage the SBM model to simulate these changes to test how well a given model can detect shifts in community structure in temporal graphs.

Barabási–Albert Model. The Barabási–Albert (BA) model is an algorithm for generating random networks with the scale-free property [5]. Networks whose degree distribution follows a power law distribution are considered to be scale-free networks. The core idea of the BA model is to produce scale-free networks with the *preferential attachment* mechanism: in real networks, new nodes tend to link to more connected nodes that already exist. Based on this insight, the BA model is generated via an iterative process: at each time step, a new node with m links is added to the existing set of n nodes where $m \leq n$ and the probability of the new node connecting to an existing node v_i is:

$$p(v_i) = \frac{d_i}{\sum_j (d_j)}$$

where d_i is the degree of node v_i and $\sum_j (d_j)$ is the sum of all node degree in the graph. The BA model can naturally generate hub structures, often observed in real-world networks. In later chapters, we use the BA model to test how well a model is able to detect shifts in a scale-free network.

Chapter 3

Related Work

In this section, we provide an overview for the temporal graph learning literature related to this thesis. We start by discussing anomaly detection on temporal graphs in Section 3.1. Then, we focus on the related work for anomaly detection on the graph snapshot level, i.e. anomalous snapshot detection, a highly relevant task in this thesis, in Section 3.2. Lastly, in Section 3.3, we discuss recent methods for temporal graph learning with an emphasis on those utilizing graph neural networks or transformers.

3.1 Anomaly Detection in Temporal Graphs

Anomaly detection is an important task with practical applications in many areas including health care, law enforcement, cyber security, finance, and more [25, 4, 161]. Anomalies are rare instances in data that deviate from the expected behavior. They are also known as outliers, outbreaks, or events in other domains. An anomaly or outlier is often defined as "an observation that differs so much from other observations as to arouse suspicion that it was generated by a different mechanism" [80]. Indeed, the definition for anomaly is vague and dependent on the specific application and context. However, the idea of measuring how much a given data point deviates from the other "normal" observations is

a key insight used by many anomaly detection methods [97, 222, 3] and is often referred to as the anomaly score.

Many complex real world relations can be represented as a graph or a network (we use these two terminology interchangeably in this work). Examples include social networks which document interactions between pairs or groups of users [225], flight networks which record plane departures and arrivals across the globe [191] and the Internet which connects computers and shares content across continents. The structure and attributes of these networks evolve constantly over time thus forming dynamic networks. As more dynamic network data is collected and monitored, it is important to identify and highlight abnormal instances from the data stream as anomalies. Concrete definitions of anomalies vary depending on the nature of the dynamic network as well as the problem domain. Some examples include faulty services in a web application [97], denial of service attacks in network systems [222], a blizzard affecting New York traffic [54] or ecological disturbances such as wildfire [31]. Early and accurate detection of such anomalies can prevent loss of money, time and even lives.

Over the past two decades, anomaly detection in dynamic graphs has received increased attention from the research community. There are many methods targeting specific sub-problems [97, 54, 207] as well as a few surveys focusing on categorizing and summarizing existing literature [161, 4, 225]. Despite significant progress made so far, there remains four key challenges associated with anomaly detection methods in dynamic graphs. First, as a result of the evolving nature of the network, *temporal reasoning* is required to compare newly arrived data with previously observed ones and identify abnormal changes that would be quantified as an anomaly. Note that the evolution of real world networks are often complex and highly domain-specific. Second, modern anomaly detection systems are deployed on large-scale networks with millions of nodes. Therefore, *scalability* must be taken into consideration when designing a novel method. Third, explaining the detected anomaly in a clear and concise way improves interpretability and becomes essential when deciding how to process or resolve an anomaly alarm after de-

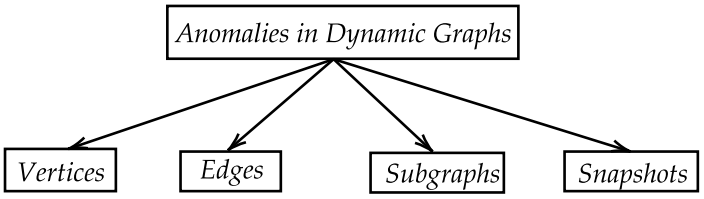


Figure 3.1: Types of anomalies in dynamic graphs.

tection. Lastly, obtaining correctly labeled anomalies might be expensive, difficult, or both. *Lack of labeled data* is a fundamental challenge during evaluation thus resulting in a large number of papers experimenting with only one or two real-world dynamic networks [97, 222, 207, 3].

Anomaly detection tasks on temporal graphs can be broadly divided into four categories: node-level, edge-level, subgraph-level and snapshot-level tasks [161], as visualized in Figure 3.1. We briefly discuss the first three categories below and provide more details on snapshot-level anomalies in Section 3.2.

Vertex-level Anomalies. The goal of anomalous vertex detection is to identify a subset of vertices within a dynamic graph whose behavior differs significantly from the others. Depending on the application, it might also be important to detect when certain nodes become anomalous. An example method named IcLEOD by Ji *et al.* [99] aims to detect nodes with abnormal evolutionary behavior with respect to their local neighborhood in a weighted dynamic graph. The core idea is to extend the egonet (all one-hop neighbors of a given node) concept to the weighted graph by introducing a novel measurement for the distance between nodes based on edge weights, named *closeness*. The nodes with high *closeness* then forms a *Corenet* of a given node of interest. Then, node-level anomalies are detected when the *Corenet* of a node undergoes anomalous changes.

Edge-level Anomalies. Anomalous edge detection aims to find a subset of edges in the network which have an irregular evolution [161]. Intuitively, this can be achieved by de-

signing a scoring function for how anomalous each edge is at each time point. Examples of abnormal edge evolution can include 1). sudden increase or decrease of edge weight and 2). the appearance of unlikely edges between two nodes that are usually disconnected. Yoon *et al.* [222] proposed ANOMRANK, a fast, online algorithm to detect two types of anomalies associated with sudden temporal changes in the graph. ANOMALYS (structure anomalies) are first defined as massive changes in the graph structure such as a node forming edges with a set of previously not connected nodes. One example of ANOMALYS is spam in email networks where a spammer sends info to many previously unrelated addresses. Then the authors defined ANOMALYW (weight anomalies) as anomalies associated with large changes in edge weights. The key insight of ANOMRANK is to model the 1st and 2nd derivatives of node scores. With a given node scoring function which represent the importance of nodes within a graph snapshot, normal behavior would evolve the node score smoothly while sudden changes induced by anomalies will be reflected in large changes in the derivatives of node scores.

Subgraph-level Anomalies. Tracking many subgraphs of interest within the evolution of a single dynamic graph requires extra effort to design a scalable solution. Different from edges and nodes, enumerating every possible subgraph within a graph snapshot can be an intractable problem. Therefore, the irregular subgraph of interest is often restricted or specified beforehand. Selecting which subgraph(s) to track can be random [54] or based on graph structures such as communities [161]. Intuitively, anomalous subgraphs are abnormal evolution of edges within a given set of nodes. Another related question is to understand what real-world factors would affect or govern the evolution of these anomalous subgraphs such as community splitting or merging, sudden increase of activity and so on. Eswaran *et al.* [54] proposed SPOTLIGHT, a randomized sketching-based approach to detect the sudden appearance or disappearance of large dense subgraphs. The motivation is that many anomalous activities involve the appearance or disappearance of a large dense subgraph. The key idea is to construct a SPOTLIGHT graph sketch which is a

vector consisting of total edge weights of K specific directed subgraphs which are tracked over time.

3.2 Anomalous Snapshot Detection

This section focus on the literature for detecting anomalous snapshots (graph-level anomalies) and provide an overview of existing methods in this area. There are two main types of anomalous snapshots: *events* and *change points*. It is often assumed that there is some unknown underlying model that governs the generative process in the dynamic graph [207]. Figure 3.2 illustrates the core difference between an event and a change point. The top of the figure shows an *event* where the graph generative model only undergoes sudden changes at one step while the bottom part shows a *change point* where the graph generative model is permanently altered after a given point. Note that in this work, we refer to the task of detecting anomalous snapshots as *change point detection* task, as standard in the literature. For our context, this task involves the detection of both types of anomalies, events and change points.

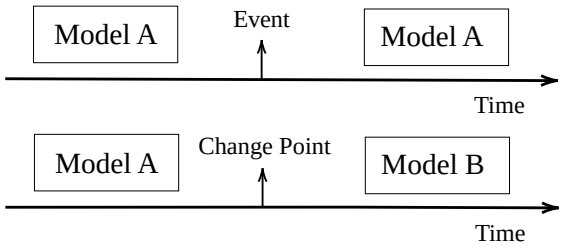


Figure 3.2: Difference between an event (top) and a change point (bottom).

3.2.1 Change Point Detection Methods

Change point detection methods focus on detecting changes in the underlying graph generative model of the temporal graph thus often employ probability based approaches. Here, we discuss example change point detection methods below.

LetoChange Peel and Clauset [155] first define the network change point detection problem. They theorized that inferring the change points requires the understanding of a structural "norm" in the dynamic graph and predicting if, when and how this norm has shifted at some point in time. We refer to the proposed method as LetoChange (similar to [207]) and there are three steps in total. First, a parametric family of probability distributions is selected as appropriate for the data. Second, two versions of the model are inferred, one representing a change of parameters at a given point inside a sliding window while the other representing the null hypothesis of no change over the window. Lastly, a statistical hypothesis test is conducted to choose if change or no change is the better fit.

EdgeMonitoring Wang *et al.* [207] proposed a novel change point detection method based on likelihood maximization, named EdgeMonitoring. EdgeMonitoring tracks a small fixed number of dyads (or node pairs) throughout the entire observed sequence of snapshots to compute a joint edge probability as the feature vector. After grouping snapshots into sliding windows, EdgeMonitoring estimates the joint edge probability with a simple consistent, and unbiased estimator for each dyad which is the proportion of snapshots in which a dyad is observed within the sliding window. To compare the signature vector across time, Wang *et al.* propose the use of Kolmogorov-Smirnov (KS) statistics as well as Euclidean distance. The threshold for detecting a change point is determined by a permutation test [156].

3.2.2 Event Detection Methods

Event detection methods focus on detecting abrupt changes when compared to the norm. Here, we discuss example event detection methods below.

Activity Vector. An early work by Idé and Kashima [97] aims to detect faulty services in a web-based computer network where each node represents a service and each edge indicates a dependency between services. They assume that the number of nodes is fixed over time while the edge weights would fluctuate. They observe that it is difficult to detect faulty services from individual nodes as the number of service calls fluctuates naturally over time. To model anomalous events on the entire dependency graph, Idé and Kashima proposed to extract the principle eigenvector $\boldsymbol{u}_t$ as an *activity vector* which summarizes the symmetric weighted adjacency matrix $\mathbf{W}_t$ at time step t. A typical (or "normal") pattern from recent activity vectors are extracted and then the dissimilarity of the current vector with the "normal" behavior is computed. The typical behavior $\hat{\boldsymbol{u}}$ is then defined as the principal left singular vector of the context matrix $\mathbf{C}$ formed by recent activity vectors. Finally, Idé and Kashima define the anomaly score as the z score, derived from the cosine similarity between the current signature vector and the typical behavior vector.

Correlation Matrix. Akoglu and Faloutsos [3] built upon the framework of [97] and extracted more complex graph properties at each snapshot as the signature vector. They focus on a weighted texting mobile network in India with over 2 million users and 50 million edges. They extract 12 specific node features at each snapshot including degree features, neighbor features, graph motifs and more. To integrate these features for change point detection, they propose the construction of the correlation matrices which contains pair-wise node features for a given timestamp. Similarly, the principal left eigenvector is used to extract the typical behavior from each correlation matrix and then compared with the feature vector at a given time t to obtain the anomaly score.

TENSORSPLAT. Koutra *et al.* [111] proposed to formulate dynamic graph data as high-order tensors for anomaly detection, named TENSORSPLAT. They showed that the "PARAFAC" decomposition method [20, 78] (also known as the CP decomposition) can be used to detect micro-clusters, rare events and changes in behaviors directly from the tensor describing the temporal graph. Each mode in the tensor $\mathcal{X}$ corresponds to a modality such as time, author, and conference and $\mathcal{X}$ describes the interactions between modalities. A specific factor matrix from the PARAFAC decomposition is then analyzed manually and qualitatively for the above tasks. One potential improvement for TENSORSPLAT would be to incorporate the output from PARAFAC decomposition into existing anomaly detection pipelines.

DELTACON. Koutra *et al.* [113] focus on answering the question of how to compare two graphs in terms of connectivity, represented by a graph similarity measure. They outlined properties that a graph similarity measure should satisfy such as the identity property, symmetric property and the zero property. Based on intuition, they further describe some properties of a graph similarity measure including a). edge importance, b). weight awareness, c). edge "Submodularity" and d). focus awareness. A novel graph similarity measure named DELTACON is proposed, the core idea is to compute pairwise node affinities in the first graph and then compare them with the ones in the second graph. In this way, the difference in the node affinity scores of the two graphs is reported as the similarity score. The anomalous snapshots are declared as the timestamps with the lowest similarity score between the current graph and the previous graph.

3.3 Temporal Graph Learning Methods

Recently, significant advances have been made for machine learning on static graphs, led by the use of *Graph Neural Networks (GNNs)* [107, 197, 29] and *Graph Transformers* [160, 114, 47], and accelerated by the availability of public datasets and standardized evaluations

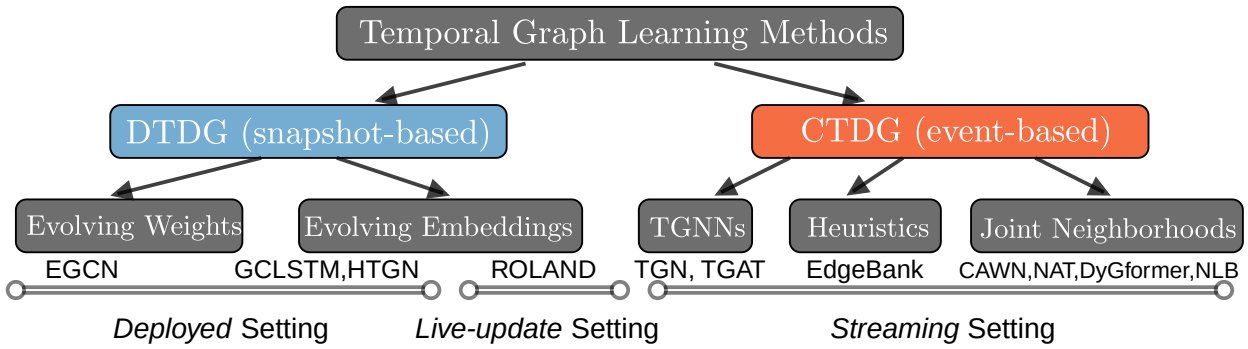


Figure 3.3: A taxonomy of TGL methods. Based on the input data type, TGL methods are often divided into DTDG methods and CTDG methods. Further categorization based on the model design can be found within both types. Lastly, TGL methods are also evaluated in three distinct evaluation settings.

protocols [88] and open sourced GNN libraries like Pytorch Geometric [55]. Similarly, a variety of machine learning approaches have been proposed for temporal graphs, often focusing on the link prediction task [206, 133, 37] or the node classification task [165].

Figure 3.3 shows a taxonomy of temporal graph learning methods. They often are categorized based on the type of dynamic graph they receive as input, namely Discrete Time Dynamic Graphs methods (see Section 3.3.1, also known as snapshot-based methods) and Continuous Time Dynamic Graph Methods (see Section 3.3.2, also known as event-based methods). DTDG methods are further divided into approaches that evolve the GNN weights over time or those that evolve the GNN node embeddings over time. In comparison, CTDG methods are broadly separated into TGNN-based methods, heuristic based methods and joint neighborhood based methods. Three distinct evaluation settings in the literature are discussed in detail in Section 3.3.3.

3.3.1 Discrete Time Dynamic Graph Methods

Methods operating on DTDGs receive a sequence of graph snapshots as input, representing the temporal graph at specific time intervals (hours, days, etc.). Therefore, DTDG

methods are designed to process entire snapshots at once (often with a graph learning model) and then utilize mechanisms to learn temporal dependencies between snapshots [131]. Evolving embedding-based methods evolve the node embeddings generated from GNNs over time by using a Recurrent Neural Network. In comparison, evolving weight-based methods such as EGCN change the weight of the GNN over time. Here, we introduce example DTDG methods below.

GCLSTM. To learn over a sequence of graph snapshots, Chen *et al.* [28] proposed a novel end-to-end ML model named Graph Convolution Network embedded Long Short-Term Memory (GC-LSTM) for the dynamic link prediction task. In this work, the LSTM act as the main framework to learn temporal dependencies between all snapshots if a temporal graph while GCN is applied on each snapshot to capture the structural dependencies between nodes. Two GCNs are used to learn the hidden state and the cell state for the LSTM and the decoder is a MLP mapping the feature at the current time back to the graph space. The design of GC-LSTM allows it to handle both link additions and link removals.

EGCN. Existing approaches often require the knowledge of a node during the entire time span of a temporal graph while real-world networks often change its node set. To address this challenge, Pareja *et al.* [153] proposed the EvolveGCN (EGCN) model which captures the dynamic of the graph sequence by using an RNN to update the weight of a GCN. In this way, The RNN regulates the GCN model parameter directly and effectively performing model adaptation. This allows node changes because the learning is performed on the model itself, rather than a specific sequence of node embeddings. Note that the GCN parameters are not trained and only computed from the RNN. Empirically, the model achieves good performance for link prediction, edge classification, and node classification on DTDGs.

HTGN. Many DTDG methods focus on learning structural and temporal dependencies in an Euclidean space thus omitting the complex and hierarchical properties that arise in real-world networks. To address this, Yang *et al.* [218] proposed a Hyperbolic Temporal Graph Network (HTGN) which utilizes the exponential capacity and hierarchical awareness of hyperbolic geometric. More specifically, HTGN incorporates a hyperbolic graph neural network and a hyperbolic gated recurrent neural network to capture the structural and temporal dependencies of a temporal graph, implicitly preserving hierarchical information. In addition, the hyperbolic temporal contextual self-attention module is used to attend to historical states while the hyberbolic temporal consistency module ensures model stability and generalization.

PyG-Temporal . PyTorch Geometric Temporal (PyG-Temporal) is an open-source Python library that combines state-of-the-art methods for neural spatiotemporal signal processing [166]. Many existing methods such as EGCN, GCLSTM, and more are implemented directly in PyG-Temporal for research. PyG-Temporal is designed with a simple and consistent API following existing geometric deep learning libraries such as Pytorch Geometric [55]. Originally, PyG-Temporal is designed for node-level regression tasks on datasets available exclusively within the framework. However, in Chapter 9, we apply PyG-Temporal models for the link prediction tasks on more publicly available datasets.

ROLAND. You *et al.* [223] proposed ROLAND as a general framework for repurposing static GNNs to DTDGs. In ROLAND, a hierarchy of node states is constructed by different GNN layers that are recurrently updated over time. The authors also introduced the novel *live-update* setting where GNNs are trained after each recently observed snapshot after making a prediction. ROLAND provides the opportunity to adapt efficient static GNN designs for the live-update setting.

3.3.2 Continuous Time Dynamic Graph Methods

Continuous Time Dynamic Graph (CTDG) methods receive a continuous stream of edges as input and make predictions over any possible timestamps. CTDG methods incorporate newly observed information into its predictions by updating its internal representation of the world. For efficiency, the stream of edges is divided into fixed-size batches while predictions are made for each batch sequentially. To incorporate the latest information, edges from each batch become available to the model once the predictions are made. Different from DTDG, CTDG has no inherent notion of graph snapshots, models often track internal representations of a node over time and sample temporal neighborhoods surrounding the node of interest for prediction. TGNN-based methods include TGN [165] and TGAT [213] are described by the framework proposed by Rossi *et al.*. Heuristic-based methods such as EdgeBank have no learnable parameters, instead relying on rules and algorithms. The last category of methods explicitly models structural information from the joint neighborhood of the node pair of an edge. Here, we describe example methods for CTDGs below.

TGAT. Xu *et al.* [213] argued that models for temporal graphs should be able to quickly generate embeddings in an inductive fashion when new nodes are encountered. The key component of the proposed Temporal Graph Attention (TGAT) layer is to combine the self-attention mechanism [196] with a novel functional time encoding technique derived from Bochner's theorem from classical harmonic analysis. In this way, a TGAT layer can efficiently learn from temporal neighborhood features as well as temporal dependencies. The functional time encoding provides a continuous functional mapping from the time domain to a vector space. The hidden vector of time then replaces positional encoding used in the self-attention mechanism.

TGN. Rossi *et al.*. [165] introduce Temporal Graph Network (TGN), a versatile and efficient framework for dynamic graphs, represented as a stream of timestamped events.

TGN leverages a combination of memory modules and graph-based operators to improve computational efficiency. Essentially, TGN is a framework that subsumes several previous models as specific instances. When making predictions for a new batch, TGN first update the memory with messages coming from previous batches to allow the model to incorporate novel information from observed batches.

CAWN. Causal Anonymous Walks (CAWs) [206] are proposed for representing temporal networks inductively to learn the laws governing the link evolution on networks such as the triadic closure law. CAWs, derived from temporal random walks, act as automatic retrievals of temporal network motifs, avoiding the need for their manual selection and counting. An anonymization strategy was also proposed to replace node identities with hitting counts from sampled walks, maintaining inductiveness and motif correlation. CAWN is a neural network model proposed to encode CAWs, paired with a CAW sampling strategy that ensures constant memory and time costs for online training and inference.

TCL. TCL [205] effectively learns dynamic node representations by capturing both temporal and topological information. It features three main components: a graph-topology-aware transformer adapted from the vanilla Transformer, a two-stream encoder that independently extracts temporal neighborhood representations of interacting nodes and models their interdependencies using a co-attentional transformer, and an optimization strategy inspired by contrastive learning. This strategy maximizes mutual information between predictive representations of future interaction nodes, enhancing robustness to noise.

NAT. In modeling temporal networks, the neighborhood of nodes provides essential structural information for interaction prediction. It is often challenging to extract this information efficiently. Luo *et al.* [133] propose the Neighborhood-Aware Temporal (NAT) network model that introduces a dictionary-type neighborhood representation for each

node. NAT records a down-sampled set of neighboring nodes as keys, enabling fast construction of structural features for joint neighborhoods. A specialized data structure called N-cache is designed to facilitate parallel access and updates on GPUs.

EdgeBank. EdgeBank [158] is a non-learnable heuristic baseline which simply memorizes previously observed edges. The surprisingly strong performance of EdgeBank in existing evaluation inspired the authors to also propose novel, more challenging, and realistic evaluation protocols for dynamic link prediction.

DyGFormer. Yu *et al.* [224] introduces a transformer-based architecture for dynamic graph learning. DyGFormer focuses on learning from nodes' historical first-hop interactions and employs a neighbor co-occurrence encoding scheme to capture correlations between source and destination nodes through their historical sequences. A patching technique was also proposed to divide each sequence into patches for the transformer, enabling effective utilization of longer histories. *DyGLib* was also presented as a library for standardizing training pipelines, extensible coding interfaces, and thorough evaluation protocols to ensure reproducible dynamic graph learning research.

NLB. Recently, Luo and Li argued that many TGL methods suffer from high computational cost and inference latency due to inefficient neighbor sampling during inference time. To resolve this challenge, they proposed No-Looking-Back (NLB), a novel TG method employing a *forward recent sampling* which bypasses the need for backtracking historical interactions. Forward recent sampling is implemented using GPU-executable hash tables for each node, enabling minimal inference latency while maintaining strong performance.

3.3.3 Evaluation Settings in TG

Existing evaluation setup can be grouped into the following categories: *streaming setting*, *deployed setting* and *live-update* setting. These settings mainly differ in how the test set information is allowed to be used for inference.

Streaming Setting. In this setting, information from the test set is only employed for updating the *memory* module in temporal graph learning methods (e.g., TGN [165]). However, no back-propagation or model update is possible based on the test set information. We consider this setting as *streaming*, since it helps in fast inference by incorporating the recent information (even from the test set), while observing the fact that retraining the model with the test data is too expensive.

Deployed Setting. In this setting, the test set information is not available for any modification to the model. This setting closely follows the standard machine learning setting with a distinct training dataset and test dataset. We refer to this setting as *deployed* to denote that after deploying the model, it is only used for inference and no updates to any part of the model is allowed.

Live-Update Setting. In this setting, information from any point in the past (including the test set information) can be used to re-train, fine-tune, or update the model. The goal of this setting is to achieve the best prediction for each timestamp $t+1$ given the historical information at all previous timestamps in $[0, ..., t]$. We consider this setting as *live-update* because the model weights can be updated lively through the incoming data, similar to the ROLAND framework [223].

Chapter 4

Laplacian Change Point Detection for Dynamic Graphs

For the first part of the thesis (Chapter 4, 5 and 6), we focus on the change point detection task. In this chapter, we first address two significant challenges associated with this task: I) how to compare graph snapshots across time, and II) how to capture temporal dependencies. Our proposed solution is Laplacian Anomaly Detection (LAD) which uses the spectrum of the Laplacian matrix of the graph structure at each snapshot to obtain low dimensional embeddings. LAD explicitly models short-term and long-term dependencies by applying two sliding windows. For synthetic experiments, LAD outperforms other state-of-the-art methods while detecting anomalous time points according to significant real-world events on three real networks. This chapter is based on our publication at KDD 2020 [93]. Related work for change point detection is discussed in Chapter 3.2.

Reproducibility: code and data for this chapter is available on Github [*].

[*] https://github.com/shenyangHuang/LAD

4.1 Introduction

Real-world problems in various domains (e.g. political science, biology, chemistry, and sociology) can be modeled as evolving networks that capture temporal relations between nodes. With the increasing availability of dynamic network data in areas such as social media, public health, and transportation, providing sophisticated methods that can identify anomalies over time is an important research direction. The goal of anomaly detection in dynamic graphs is to identify different types of time-varying anomalies that significantly deviate from the "normal" or "expected" behavior of the underlying graph distribution. We provide some motivating examples below.

In a transportation network, traffic volumes can be represented as edge weights in isolated network snapshots that are taken at various times of the day. When a traffic accident occurs, a drastic change in that section of the network will likely follow (e.g. decreasing amount of traffic). On the other hand, cyberbullying, terrorist attack planning, and fraud information dissemination [225] can all be seen as cases of anomalies in a temporal social graph. Identifying these anomalies accurately and rapidly can result in positive and significant real-world impact. Lastly, complex clinical information can also be represented as a dynamic graph. Patients, symptoms, and treatments can be represented as vertices in a heterogeneous network. Detecting anomalies in such networks can discover critical scenarios where errors or abnormal patient conditions arise [30].

In this work, we aim to tackle two major challenges for change point detection:

1). How to compare graph snapshots across time. To compare graph level changes over time, a summary of the network in the form of a low dimensional representation is often used [225]. We propose to use the singular values of the Laplacian matrix (Laplacian spectrum) because it represents a global view of the graph snapshot and of its connection to graph connectivity and low-rank approximation. Due to these connections, we show that LAD can detect a wide range of graph changes (i.e. community structure and average edge weight) in dynamic graphs in Section 4.5.

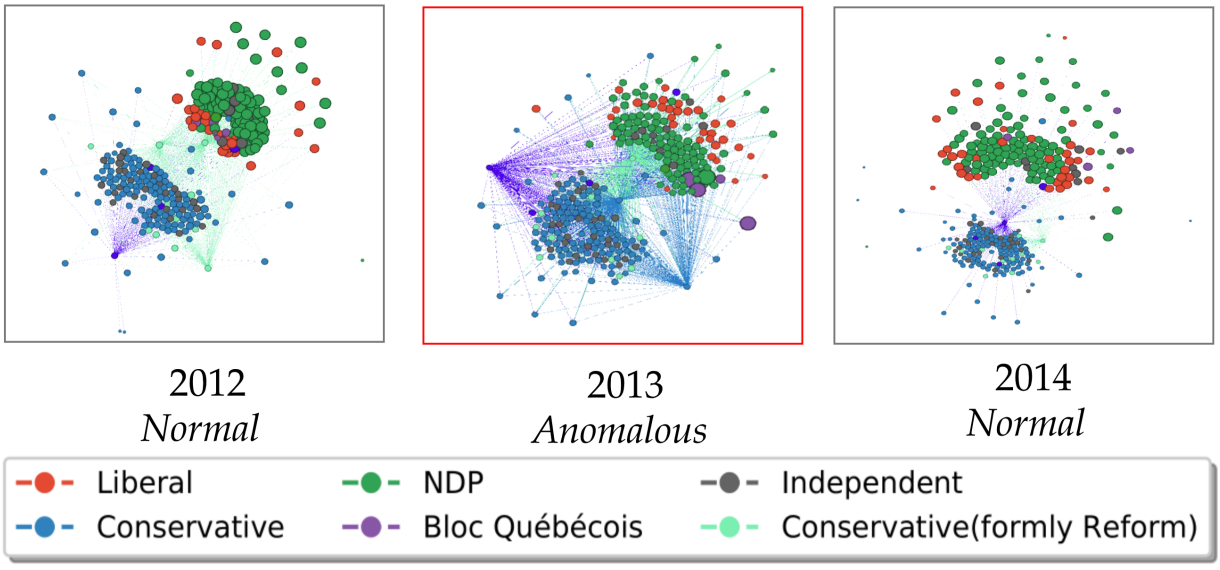


Figure 4.1: LAD detects changes in Canadian Member of Parliament voting patterns. LAD identified 2013 as anomalous due to abnormal amount of edges between political parties.

2). How to capture temporal dependencies. In practice, graphs can undergo abrupt changes at any given time step [58]. These sudden changes can be identified through a short-term sliding window [3, 97]. However, points in time that signal a change in graph pattern for a long duration of time can also have high significance [161]. In particular, it is possible for the underlying graph generation model to evolve [155, 207]. Effective detection of these points of change would require the model to reason with the graph behaviors beyond the most recent ones. We propose to use two context windows that explicitly compare the current graph structure with the typical behaviors from both short-term and long-term perspectives.

Figure 4.1 shows the anomalous snapshot (inside the red box) detected by our method in the Canadian parliament voting network. Nodes are colored by political parties and node sizes are weighted using PageRank [148]. Our method detected anomalous cross-party interaction in 2013. We summarize features of LAD and other alternative methods in Table 4.1. Note that LAD is the only approach that satisfies all the desired properties.

Method	event	change point	scalable	evolving # nodes	node permutation inv.
LAD (ours)	✔	✔	✔	✔	✔
Activity vector [97]	✔			✔	✔
TENSORSPLAT [111]	✔	✔			✔
EdgeMonitoring [207]		✔	✔		

Table 4.1: LAD satisfies all the desired properties while other change point detection methods lack one or more.

4.2 Methodology

We propose a new spectral anomaly detection method for dynamic graphs: Laplacian Anomaly Detection (LAD). The core idea of LAD is to detect high level graph changes from low dimensional embeddings (called signature vectors). The "typical" or "normal" behavior of the graph can be extracted from a stream of signature vectors based on both short-term and long-term dependencies. Now, we can compare the deviation of the current signature vector from the normal behavior. Figure 4.2 shows the flowchart of our method.

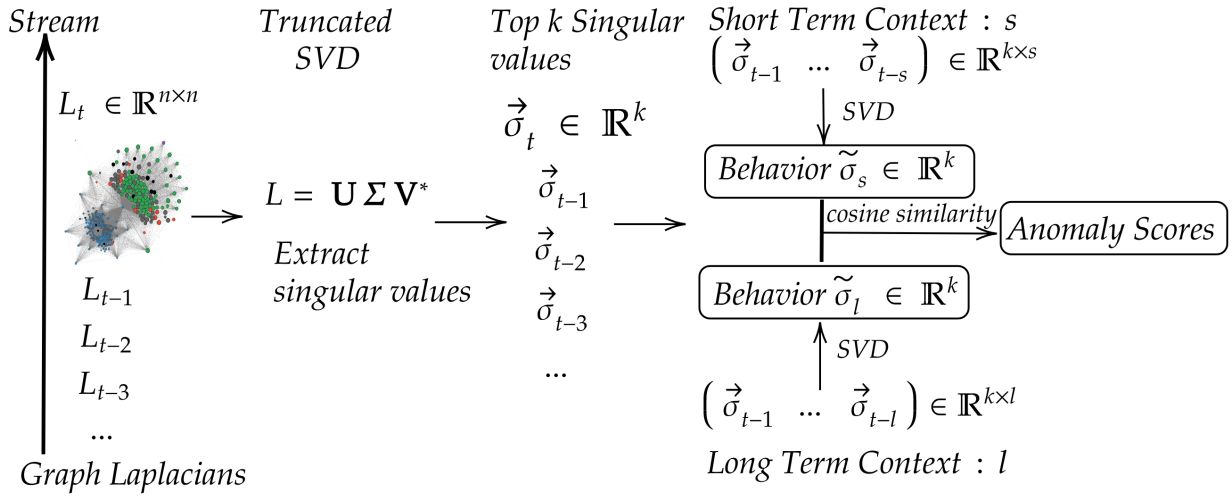


Figure 4.2: The flow chart of LAD.

4.2.1 Laplacian Spectrum

We define the (unnormalized) Laplacian matrix $\mathbf{L}_t$ as $\mathbf{L}_t = \mathbf{D}_t - \mathbf{A}_t$ where $\mathbf{D}_t$ is the diagonal degree matrix and $\mathbf{A}_t$ is the adjacency matrix of $\mathcal{G}_t$. In this work, we choose the singular values obtained through Singular Value Decomposition (SVD) [67] of the Laplacian matrix as graph embeddings for each snapshot. This is motivated by the fact that singular values 1). are related to the Laplacian spectrum, 2). encodes the compression loss of low-rank approximations of the Laplacian matrix, 3). are node permutation invariant and 4). can be efficiently computed in real-world sparse matrices. Figure 4.3 shows the visualization of the Laplacian spectrum and the corresponding anomaly scores detected by LAD for the Senate co-sponsorship network.

First, it is known that the singular values of a positive symmetric matrix coincide with its eigenvalues. The Laplacian matrix is symmetric and positive semi-definite for an undirected weighted graph [200]. The eigenvalues of the Laplacian matrix capture the fundamental structural properties of the corresponding graph. This property has been extensively leveraged in many fields such as randomized algorithms, combinatorial optimization problems, and machine learning [231], and the field of spectral graph theory [33] is dedicated to the study of graph Laplacian matrices. As an illustration, the multiplicity k of the eigenvalue 0 of $\mathbf{L}$ equals the number of connected components in the graph [200]. In addition, the Laplacian spectrum is related to many other structural properties of the graph such as the degree sequence, number of connected components, diameter, vertex connectivity, and more [231].

Second, it is well known that the truncated SVD gives the best low-rank approximation of a matrix with respect to both the Frobenius norm and the 2-norm (see Eckart-Young theorem [50]). More precisely, the $(k+1)$th singular value σ_{k+1} of a matrix corresponds to the reconstruction error of the best rank k approximation measured in the 2-norm. From this insight, we know that the (ordered) singular spectrum $\sigma_1, \sigma_2, ..., \sigma_r$ encodes rich information regarding the reconstruction loss that would occur for different levels of low-rank approximations. Truncated SVD is often used as a powerful compres-

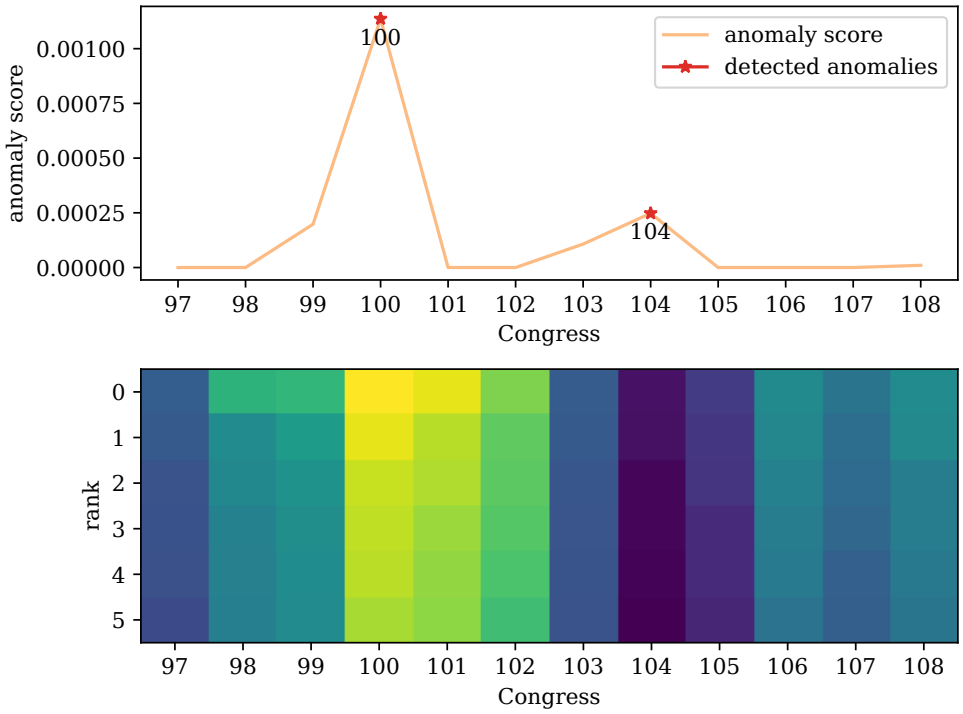


Figure 4.3: The changes in the graph spectrum of the Senate co-sponsorship network is captured by LAD. The LAD scores (top) and the top 6 singular values (bottom) are aligned at each timestamp. The warmer the color, the higher the intensity.

sion tool for images [168], videos [10] and audios [227]. Thus, capturing the singular spectrum can be seen as an alternative compression-based anomaly detection technique as categorized by Ranshous et al. [161]. Intuitively, huge fluctuations in the singular values of the Laplacian matrix reflect drastic changes to the global graph structure. In an undirected graph where the Laplacian singular values coincide with the eigenvalues, the decrease in the number of zero singular values reflects the decrease in the number of connected components.

Third, change point detection can be viewed as a binary graph classification problem. Being node permutation invariant is one of the most desirable properties for a graph learning method [214]. However, preserving the same node ordering for each graph snapshot might not be feasible in many practical settings. Since applying row or column permutations on the Laplacian matrix has no effect on the singular values, LAD is node

permutation invariant. In this way, LAD can be applied to a broader range of scenarios than methods relying on a consistent node ordering such as EdgeMonitoring [207].

Lastly, singular values can be efficiently obtained through sparse computations [1]. As many real-world graphs are sparse, the proposed graph embedding can be scaled to large datasets. In addition, depending on the computational budget, one can compute the top k singular values through truncated SVD with much less computational cost when compared to the full SVD. In Section 4.3, we show that for some real-world datasets, it is often enough to compute the top k singular values. We also explain the computational complexity of LAD in Section 4.2.5.

4.2.2 Characterizing Normal Behavior

Identifying a normal or typical behavior from a temporal window is often an integral part of change detection. Similar to [3, 97], we compute a "typical" or "normal" behavior vector from the previous l singular spectrums where l is the sliding window size. First, we perform *L2* normalization on the Laplacian spectrums seen so far $\boldsymbol{\sigma}_0, \ldots, \boldsymbol{\sigma}_t$ to obtain unit vectors. Next, a context matrix $\mathbf{C}$ is constructed:

$$\mathbf{C} = \begin{pmatrix} | & | & & | \\ \boldsymbol{\sigma}_{t-l-1} & \boldsymbol{\sigma}_{t-l-2} & \ldots & \boldsymbol{\sigma}_{t-1} \\ | & | & & | \end{pmatrix} \in \mathbb{R}^{n \times l} \tag{4.1}$$

where n is the length of the signature vector. We compute the left singular vector of $\mathbf{C}$ with SVD to obtain the normal behavior vector $\tilde{\sigma}_t$. In literature, this is often considered as a weighted average vector from the sliding window [3].

4.2.3 Two Perspectives

Different from [3, 97], we propose to compare the current signature vector with the typical behavior from two independent sliding windows: a short-term window and a long-term

window. The short-term window encodes information from the most recent trend and captures abrupt changes in the overall graph structure. Depending on the application, the length of the short-term window can be adjusted to best reflect an appropriate time scale. On the other hand, a long-term window is designed to capture larger-scale and more gradual trends in the dynamic graph. For example, for the UCI Message dataset, the short-term context can be monitoring weekly change and the long-term context can be monitoring a biweekly change.

4.2.4 Anomalous Score Computation

After capturing the normal behavior, one can then define a scoring function to measure the difference between the current signature vector and the expected or normal one. In this work, we use the same anomaly score as introduced in [97, 3], namely the Z score. Let $\tilde{\sigma}_t$ be the normal behavior vector and $\boldsymbol{\sigma}_t$ be the Laplacian spectrum at the current step. As mentioned in Section 4.2.2, both $\tilde{\sigma}_t$ and $\boldsymbol{\sigma}_t$ are normalized to unit vectors, then the Z score is computed as:

$$Z = 1 - \frac{\boldsymbol{\sigma}_t^\top \tilde{\sigma}_t}{\|\boldsymbol{\sigma}_t\|_2 \|\tilde{\sigma}_t\|_2} = 1 - \boldsymbol{\sigma}_t^\top \tilde{\sigma}_t = 1 - \cos\theta, \tag{4.2}$$

where $\cos\theta$ is the cosine similarity between $\boldsymbol{u}$ and $\boldsymbol{v}$. Essentially the Z scores become closer to 1 when the current spectrum is very dissimilar to the norm thus signaling an anomalous point.

Let s and l be the sizes of the short-term and long-term sliding windows. Analogous to Section 4.2.2, one can obtain two different normal behavior vectors $\tilde{\sigma}_s$ and $\tilde{\sigma}_l$. $\tilde{\sigma}_s$ encodes the expected norm within a few time steps while $\tilde{\sigma}_l$ captures the normal network activity within a larger time span. One can then compute the short-term and long-term anomaly scores Z_s and Z_l based on cosine similarity. To best aggregate these two perspectives, we take $Z_t = max(Z_s, Z_l)$ to decide if the current graph is more anomalous in abrupt or gradual changes.

Now having a sequence of anomaly scores $Z_1, \ldots, Z_t$, how to best select the change points based on these scores? Different than [97, 3], we choose the points that have the largest increase in anomaly score when compared to the previous time step. Therefore, we have the final anomaly score $Z_t^* = min(Z_t - Z_{t-1}, 0)$. The points with the largest Z^* are then selected as anomalies.

4.2.5 Computational Complexity

Given a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, the computational complexity for full SVD is $\mathcal{O}(m^2 n)$ when $m \leq n$. For a truncated rank k SVD, the complexity is $\mathcal{O}(mnk)$. Halko et al. [71] showed that randomization offers a powerful tool for performing low-rank matrix approximation. Randomized SVD more efficiently utilizes computational architectures thus achieving $\mathcal{O}(mn \log(k))$ cost. At the same time, Berry [13] presented strong numerical methods for computing SVD in large sparse matrices on a multiprocessor architecture. As many real-world networks are sparse, sparse SVD can achieve significant computational savings. In this work, we use the Scipy [198] sparse SVD implementation in Python.

4.3 Experiments

In this section, we perform extensive experiments on different types of synthetic and real world datasets to validate the effectiveness of the LAD framework.

Evaluation Setting. For quantitative evaluation of a change point detection method, we use Hits at n ($H@n$) metric which reports the number of identified significant anomalies out of the top n most anomalous points. In synthetic experiments, we use the ground truth labels in the generation process for evaluation. In real world datasets, we treat the well-known anomalous times steps as ground truth anomalies which should be detected by a given method.

Contenders and Baselines. We compare LAD with the following alternative methods. The same short-term and long-term window sizes (s and l) are used if applicable. The short-term and long-term anomaly scores are both shown in Figure 4.4,4.5,4.6,4.7,4.8. Note that for all experiments, the startup period $(0, \ldots, l)$ is set to have an anomaly score of 0 because we assume these points are not change points.

- **Activity vector.** We refer to the method proposed by Idé et al. [97] as "activity vector" based methods. Idé et al. used the principal eigenvector of the adjacency matrix (namely the activity vector) instead of our proposed Laplacian spectrum. According to the original work, only a short-term context window is considered.

- **TENSORSPLAT.** Koutra et al. [111] proposed to view the temporal graph as a tensor and then perform PARAFAC decomposition to obtain low dimensional factors that group similar entities or timestamps together (we use CP rank of 30 for all experiments). The original paper proposed to use clustering on the factors for change detection. However, the clustering algorithm is not specified. We use the well-known Local Outlier Factor (LOF) [19] approach along with the TENSORSPLAT framework.

- **EdgeMonitoring.** Wang et al. [207] proposed the EdgeMonitoring approach and used joint edge probabilities as the "feature vector" while modeling network evolution as a first-order Markov process.

Interpreting Results. It is often difficult to interpret the anomaly score of a given change point detection method as the task inherently demands direct comparison between global graph structures over time. As network characteristics vary drastically across domains [21], it is important to design metrics that help us understand the correlation between anomaly scores and well-known graph properties. In this work,we identify anomalies in specific graph properties and compare them to the ones predicted by LAD. We choose the outlier score y as follows:

$$y = \frac{|\alpha_t - \alpha_{avg}|}{\alpha_{std}}, \tag{4.3}$$

Order	Time Point	Generative SBM Model
0	0	$\mathbf{N_c} = \mathbf{4},\ p_{in} = 0.25,\ p_{ex} = 0.05$
1	16	$\mathbf{N_c} = \mathbf{10},\ p_{in} = 0.25,\ p_{ex} = 0.05$
2	31	$\mathbf{N_c} = \mathbf{2},\ p_{in} = 0.5,\ p_{ex} = 0.05$
3	61	$\mathbf{N_c} = \mathbf{4},\ p_{in} = 0.25,\ p_{ex} = 0.05$
4	76	$\mathbf{N_c} = \mathbf{10},\ p_{in} = 0.25,\ p_{ex} = 0.05$
5	91	$\mathbf{N_c} = \mathbf{2},\ p_{in} = 0.5,\ p_{ex} = 0.05$
6	106	$\mathbf{N_c} = \mathbf{4},\ p_{in} = 0.25,\ p_{ex} = 0.05$
7	136	$\mathbf{N_c} = \mathbf{10},\ p_{in} = 0.25,\ p_{ex} = 0.05$

Table 4.2: Change points in the pure setting when the number of blocks (N_c) change.

where α_{avg}, α_{std} are the average and standard deviation of α computed from a moving window. We select the moving window size to correspond with the short-term window size s. Then we compute the Spearman rank correlation [237] to understand the statistical dependence between two ranked variables. We observe that LAD is not relying on one particular graph statistics but rather on the most important aspects of the dynamic graph of interest.

4.3.1 Synthetic Experiments

To demonstrate the performance of LAD, we design three controlled experiments. For these synthetic experiments, we use data generated from the Stochastic Block Model (SBM) [83]. The number of communities k as well as the number of nodes within each community can be specified through a size vector $\boldsymbol{s} \in \mathbb{R}^k$. In addition, the inter-community and intra-community connectivity for each block can be directly encoded in a symmetric probability matrix $P \in \mathbb{R}^{k \times k}$. For all experiments, we set the short-term and the long-term window to be 5 and 10 time steps respectively, and use the entire Laplacian spectrum in LAD.

Pure Setting. Here, we only introduce change points, where the adjustments in community structure persist until the next change point is reached. We generate a temporal net-

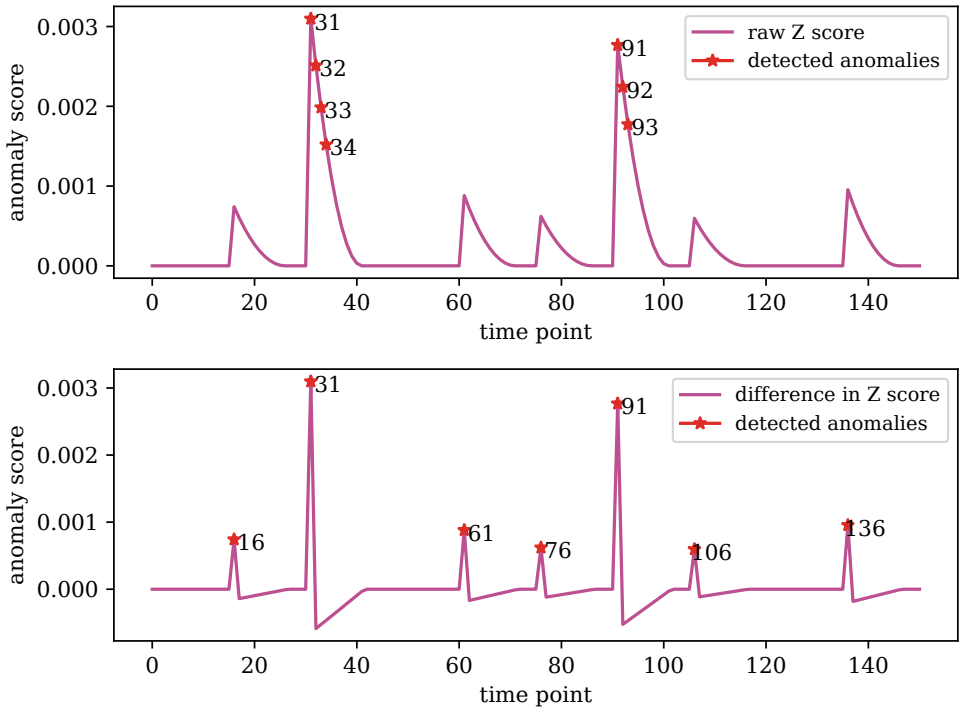


Figure 4.4: The predicted anomaly scores of LAD for Table 4.2. The top is the raw Z score while the bottom is the Z^* scores.

work with 151 time points where each snapshot is produced through SBM parametrized by $\boldsymbol{s}$ and P. There are always 500 nodes per snapshot and the community change is described in Table 4.2. Here N_c represents the number of equal-sized communities in the snapshot, and p_{in}, p_{ex} denotes the internal and external community connectivity probability respectively. We set the continuity rate [207] to be 0 for change points and 1.0 elsewhere for the pure setting.

The anomaly score predicted by LAD can be seen in Figure 4.4. The top 7 most anomalous points correspond to the 7 ground truth points in Table 4.2 while the other points have extremely low anomaly scores. This supports the empirical observation that the Laplacian spectrum is indeed sensitive to changes in community structure. EdgeMonitoring and LAD both achieves perfect precision as both can reason with gradual changes over time. In contrast, both TENSORSPLAT and Activity vector can only recover some

Graph Property	Spearman Rank Correlation
# of connected components	**17.0**%
Transitivity	11.8%
# of edges	7.5%
Average degree per node	15.9%

Table 4.3: Spearman rank correlation between LAD and various graph properties.

change points. As the activity vector uses the principal eigenvector of the adjacency matrix, it is unable to detect community changes as well as the Laplacian spectrum used in LAD.

By examining Spearman rank correlations in Table 4.3, we observe that LAD predictions are most strongly correlated with the number of connected components while still having positive correlations with other network properties such as transitivity. As LAD captures high level graph structures, it is sensitive to the important properties in the dynamic graph of interest while not dependent on any single property.

Note that when using the raw Z scores in Figure 4.4, we observe declines in anomaly score after the change point. In a sliding window that contains graph snapshots from the previous graph generative process and the current one, the normal behavior vector is computed based on a mix of graphs from two generative processes thus leading to the observed declining anomalous scores after the change point. In comparison, using Z^* scores makes LAD more robust under different choices of sliding windows. The negative Z^* scores are only shown for illustrative purposes In Figure 4.4, in other figures, they are set to 0.

Hybrid Setting. To study the effectiveness of LAD in both change point and event detection, we generate a synthetic experiment with SBM where these two types of changes both exist. We generate events by strengthening the inter-community connectivity for that time point (subsequent points are not affected). These events can correspond to sudden increased collaborations between usually separated communities such as political

Time Point	Type	Generative SBM Model
0	start point	$N_c = 4,\ p_{in} = 0.25,\ p_{ex} = 0.05$
16	event	$N_c = 4,\ p_{in} = 0.25,\ \mathbf{p_{ex} = 0.15}$
31	change point	$\mathbf{N_c = 10},\ p_{in} = 0.25,\ p_{ex} = 0.05$
61	event	$N_c = 10,\ p_{in} = 0.25,\ \mathbf{p_{ex} = 0.15}$
76	change point	$\mathbf{N_c = 2},\ p_{in} = 0.5,\ p_{ex} = 0.05$
91	event	$N_c = 2,\ p_{in} = 0.5,\ \mathbf{p_{ex} = 0.15}$
106	change point	$\mathbf{N_c = 4},\ p_{in} = 0.25,\ p_{ex} = 0.05$
136	event	$N_c = 4,\ p_{in} = 0.25,\ \mathbf{p_{ex} = 0.15}$

Table 4.4: Model changes in the Hybrid setting where we have combination of event and change points.

Metric	Hits @ 7			Hits @ 10	Hits @ 2
Dataset	Pure	Hybrid	Resampled	UCI	Senate
LAD (ours)	**100%**	**100%**	**100%**	**50%**	**100%**
EdgeMonitoring [207]	**100%**	**100%**	0%	0%	**100%**
Activity vector [97]	71.4%	0%	0%	**50%**	50%
TENSORSPLAT [111]	28.5%	14.2%	57.1%	0%	50%

Table 4.5: LAD consistently finds significant anomalies across different datasets and outperforms alternative approaches. The hybrid experiment refers to the event and change point detection experiment.

parties. The details regarding the generative process can be seen in Table 4.4. We set the continuity rate [207] to be 0 for change points and 0.9 elsewhere for the hybrid setting. From Figure 4.5, we observe that LAD is able to perfectly recover all events and change points.

Resampled Setting. In this setting, we use a constant continuity rate of 0 for all time steps, i.e. the graph is resampled from the generative process at each step. In real-world graphs, majority of the edges might not persist over consecutive time steps but rather be determined by the underlying community structure such as a club or class. For the generation parameters and change points' details, we use an identical setup as the Hybrid Setting. The complete resampling from the SBM model can be considered as a node-

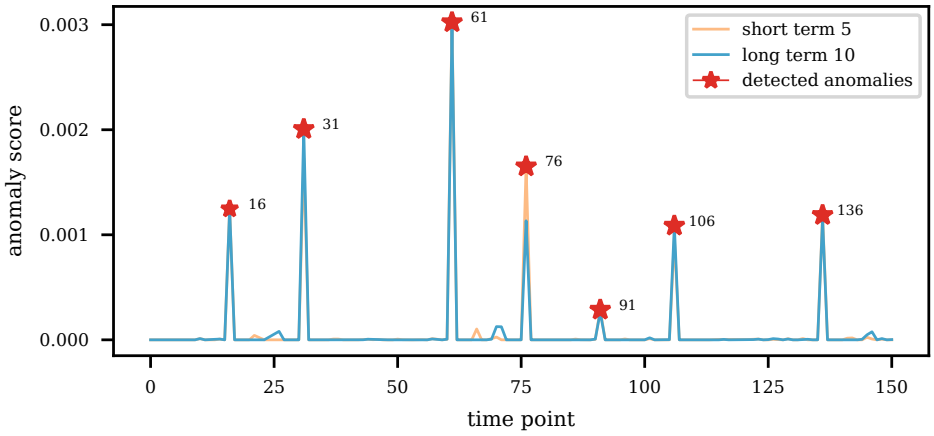


Figure 4.5: The anomalies scores of LAD for the experiment shown in Table 4.4. Both events and change points are captured.

level permutation within each community. As discussed in Section 4.2, the eigenvalues of the Laplacian is node permutation invariant. Indeed, LAD outperforms all baselines. In comparison, EdgeMonitoring selects specific pairs of node pairs to track over time thus it is susceptible to node permutation and results in poor performance.

4.3.2 Real-world Experiments

Here, we evaluate the performance of LAD on two real-world benchmark datasets, as well as an original dataset. we report the performance of LAD and all baselines in Table 4.5. For UCI Message and Senate co-sponsorship experiments, LAD is able to achieve strong performance using only the top 6 eigenvalues. For the Canadian bill voting network, we report the LAD performance with the top 338 eigenvalues [†].

UCI Message. The UCI Message dataset is a directed and weighted network based on an online community of students at the University of California, Irvine. Each node represents a user and each edge encodes a message interaction from one user to another. The

[†]338 is the number of seats in the House of Commons of Canada

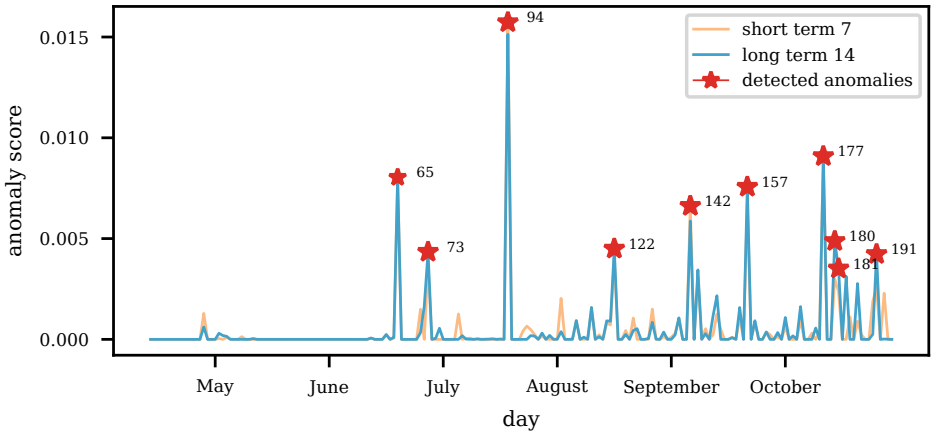


Figure 4.6: LAD correctly detects the end of the university spring term and one day before the start of the fall term in the UCI message dataset. The above shows a visualization of both the short-term and long-term Z^* scores.

weight of each edge represents the number of characters sent in the message. When an user account is created, a self-edge with unit weight is added. A total of 1,899 users was recorded. The network data covers the period from April to October 2004 and spans 196 days.

In this work, we treat each day as an individual time point. There are 59,835 total messages sent across all time stamps and 20,296 unique messages. To ensure privacy protection, all individual identifiers such as usernames, email, and IP addresses were removed thus we use the dataset description provided in [150] to find significant events in the dataset. We select the short-term window to be 7 days as suggested by Panzarasa et al. The long-term window is then selected to be 14 days or two weeks.

Figure 4.6 shows the Z^* scores predicted from both the short-term and long-term window (with top 6 eigenvalues). Panzarasa et al. mentioned that day 65[‡] is the end of the spring term and day 158 is the start of the fall term. Day 65 is correctly predicted by LAD and activity vector while the top 10 predictions from EdgeMonitoring and TEN-

[‡]we set the index to start at 0

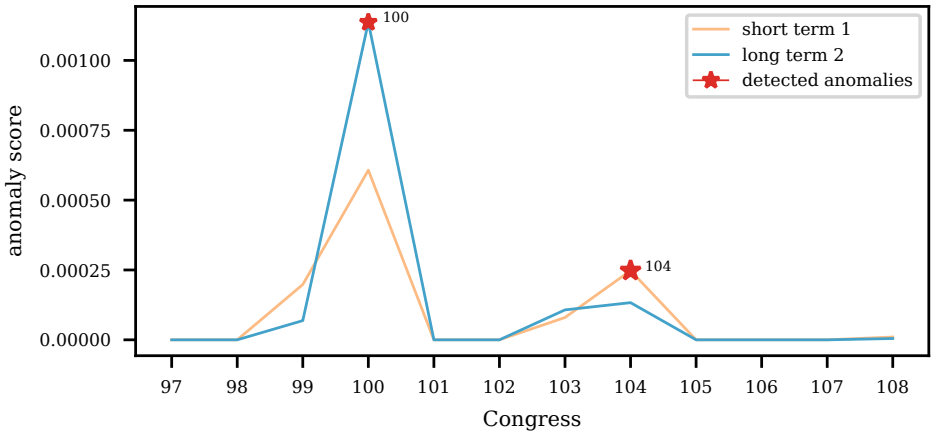


Figure 4.7: LAD correctly detects the 100th and 104th congress as the top 2 most anomalous points.

SORSPLAT do not include either day. However, LAD predicts day 157 as anomalous which corresponds to the day prior to the start of the fall term. It is likely that anomalous message behaviors are exhibited before school starts. As the edge weight (number of characters in messages) shows important connections between users in social networks, the Z scores has strongest correlation to the average edge weight per snapshot in this dataset.

Senate Co-sponsorship Network. Senate co-sponsorship network [56] examines social connections between legislators from their co-sponsorship relations on bills during the 93rd-108th Congress. An edge is formed between two congresspersons if they are cosponsors of a bill. Bills are grouped into temporal snapshots biannually (time frame for each graph) and co-sponsors on a bill form a clique. Similar to [207], we start from the 97th Congress as full amendments data is available from there onward. Fowler [56] pointed out that the 104th Congress corresponds to "a Republican Revolution" which "caused a dramatic change in the partisan and seniority compositions." Wang et al. also stated that the 104th Congress has the lowest clustering coefficient and thus can be seen as a

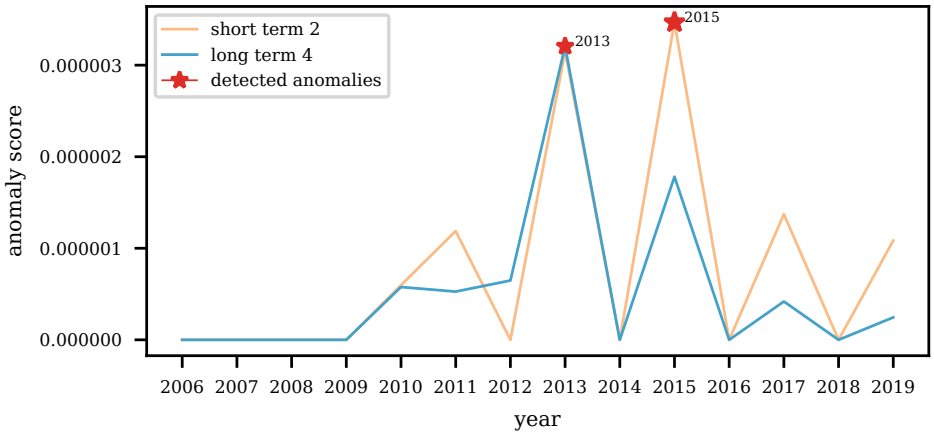


Figure 4.8: LAD predicts years 2013 and 2015 as anomalous years for the Canadian bill voting network.

low point in collaboration while the 100th Congress has the highest clustering coefficient which signals significant collaboration.

From Figure 4.7, it is clear that LAD can easily detect the above change points (using only the top 6 singular values). Variations of LAD that only utilizes the short-term or the long-term window are able to identify both points too. Wang et al. mentioned that EdgeMonitoring [207] and LetoChange [155] are able to detect both change points while DeltaCon [112] only predicts the 104th Congress.

The activity vector method requires full SVD which is computationally expensive and it is only able to detect the 100th Congress. The anomaly scores output by the activity vector method have similar magnitude thus making it difficult to identify change points. However, if we augment the activity vector method to use LAD pipeline and aggregates two sliding windows, it is able to correctly predict both change points. This shows that aggregating the output from two sliding windows can improve empirical performance with a different graph embedding technique.

Canadian Bill Voting Network. To understand the temporal change in the Canadian Parliament environment, we extracted open information[§] to form the Canadian Parliament bill voting network. The Canadian Parliament consists of 338 Members of Parliament (MPs), each representing an electoral district, who are elected for four years and can be re-elected [36]. In the included time frame from 2006 to 2019, the increase in the number of electoral seats resulted in parliaments with different amounts of MPs; since 2015 the House of Commons has grown from 308 seats to 338. Naturally, the number of nodes in the network fluctuates from year to year. 2 years and 4 years are used as the duration for the short-term and long-term windows respectively. We consider the MPs that voted yes for a bill to have a positive relationship with the sponsor. In this way, a directed edge is then formed from a voter MP u to the sponsor MP v in a given graph snapshot. Each edge is then weighted by the number of times that u voted positively for v.

LAD detects 2013 and 2015 as two significant change points. Figure 4.8 shows the Z^* scores predicted by LAD. 2013 is often considered as the year where cross-party cohesion against Conservatives begins to decline. Prior to 2013, the Liberal and New Democratic Parties had more unity in voting patterns. As a new party leader (Justin Trudeau) is elected in the Liberal party in 2013, changes in voting patterns are observed [136]. In 2015, the House of Commons increased the number of constituencies from 308 to 338. Equally, on October 19th 2015 an election took place and the Liberal party won an additional 148 seats, with a total of 184 seats forming a majority government led by Justin Trudeau. Prior to 2015, the Liberal party was divided and however during this election things changed and literature shows the unified campaign at a local level from different members of parliament across the country [38].

[§]extracted from https://openparliament.ca/

4.4 Conclusion

In this chapter, we proposed a novel spectral-based method capable of both change point and event detection, called LAD. The core idea of LAD is to summarize entire graph snapshots into low dimensional embeddings through the singular values of the graph Laplacian. Different than previous approaches, LAD explicitly models the short-term and the long-term behavior of the dynamic graph and aggregates both perspectives. When compared to existing methods, LAD outperforms all alternative methods on synthetic experiments and LAD finds well-known events in three real networks.

Chapter 5

Fast and Attributed Change Detection on Dynamic Graphs with Density of States

Current solutions for change point detection do not scale well to large real-world graphs, lack robustness to large amounts of node additions/deletions, and overlook changes in node attributes. For example, LAD requires the computation of the Laplacian spectrum which can be prohibitively expensive for large graphs. Therefore, in this chapter, we focus on addressing these limitations. To this end, we propose a novel spectral method: Scalable Change Point Detection (SCPD). SCPD generates an embedding for each graph snapshot by efficiently approximating the distribution of the Laplacian spectrum at each step. SCPD can also capture shifts in node attributes by tracking correlations between attributes and eigenvectors. Through extensive experiments using synthetic and real-world data, we show that SCPD (a) achieves state-of-the-art performance, (b) is significantly faster than the state-of-the-art methods and can easily process millions of edges in a few CPU minutes, (c) can effectively tackle a large number of node attributes, additions or deletions and (d) discovers interesting events in large real-world graphs. This chapter is based on our publication at PAKDD 2023 [92].

Reproducibility: code for this chapter is available on Github [*].

5.1 Introduction

Anomaly detection is one of the fundamental tasks in analyzing dynamic graphs [155, 53], with applications ranging from detecting disruptions in traffic networks, analyzing shifts in political environments, and identifying abnormal events in communication networks. However, detecting anomalies in dynamic graphs offers several challenges: real world graphs are often very large, their size can drastically evolve over time (e.g. nodes appearing and disappearing in social network graphs where nodes represent users) and complex information is associated with nodes in the graph (e.g., profile of users summarized as a set of attributes for each node).

Prior work on change point detection is limited by one or more of the following issues. *1). Lack of scalability*: modern networks often contain millions of edges and nodes, thus computationally intensive algorithms [93, 111] can be difficult to apply on graphs with more than hundreds of nodes. *2). Overlooking attributes*: many networks also contain a diverse set of node attributes that evolve over time. No prior work has considered the evolution of node attributes and their relation with the graph structure. *3). Difficulty with evolving sizes*: real networks grow over time with new nodes often forming a large portion of the network. Methods such as [54, 207] track a fixed set of nodes sampled from the initial time step and are thus limited to detecting changes happening within the initial set of nodes. Other approaches such as [93, 97] summarize each snapshot with a vector dependent on the size of the snapshot. Therefore, as the graph grows, truncation on the summary vector is required to ensure a uniform vector size for all snapshots.

To address the above limitations, we propose Scalable Change Point Detection (SCPD), a novel change point detection method which detects both structural and node attribute anomalies in dynamic graphs. SCPD utilizes the distribution of eigenvalues (also known

[*] https://github.com/shenyangHuang/SCPD

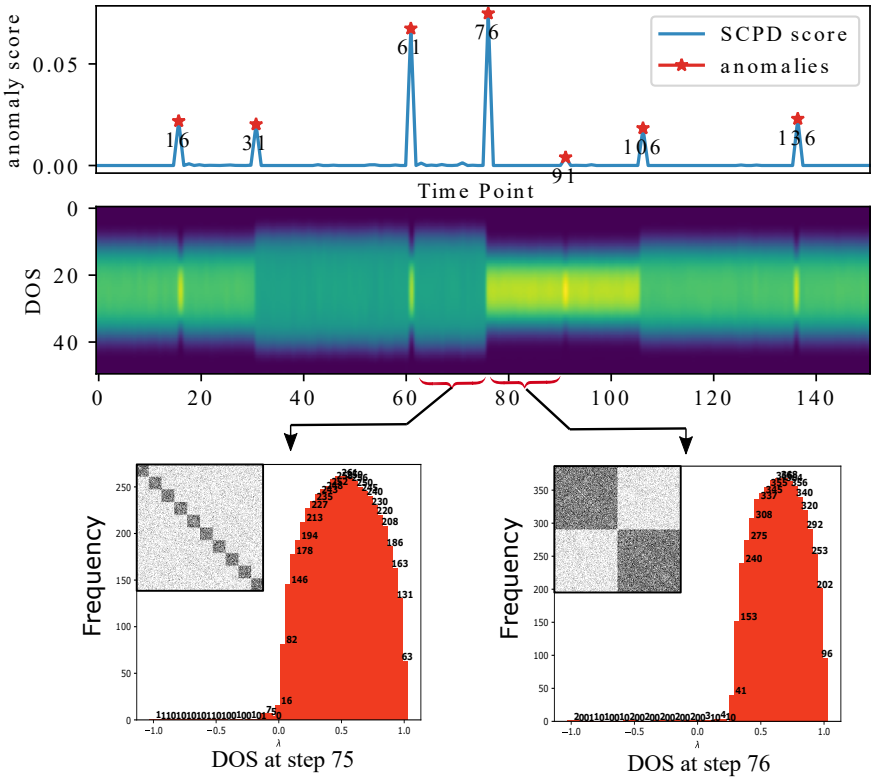


Figure 5.1: SCPD utilizes the spectral density (approximated by Density of States (DOS)) to summarize the graph at each time point, change in DOS often indicates a change in graph distribution. The DOS becomes skewed after the number of communities decreases from ten to two in the SBM [83] hybrid experiment (see Section 5.5). The DOS is plotted for step 75 and 76 while the inset plots show the adjacency matrix of the graph.

as the spectral density) of the Laplacian matrix as a low dimensional embedding of each graph snapshot. As change points induce a shift in graph distribution, they also cause changes in the spectral density. We leverage the Density of States (DOS) [45] framework to efficiently approximate the spectral density, allowing SCPD to scale to dynamic graphs with millions of nodes. Figure 5.1 illustrates the key idea of SCPD: to discretize the spectral density, the range of eigenvalues is divided into k bins and the number of eigenvalues within each bin is computed. As such, the number of bins k is not dependent on the size of the network. Therefore, SCPD can easily adapt to the evolving size of a dynamic graph. The main characteristics of SCPD are:

Table 5.1: Properties of existing methods compared to SCPD.

Method \ Property	Event	Change Point	Scalable	Evolving Size	Weights	Attributes
Activity vector [97]	✔		✔	✔	✔	
TENSORSPLAT [111]	✔				✔	
EdgeMonitoring [207]		✔	✔			
SPOTLIGHT [54]	✔		✔		✔	
LAD [93]	✔	✔		✔	✔	
SCPD [this paper]	✔	✔	✔	✔	✔	✔

- **Accurate:** We show that SCPD achieves state-of-the-art performance in extensive synthetic experiments and can identify a number of major wars from the co-authorship network MAG-History of the History research community (while existing methods fail to adapt to the evolving size of this network).

- **Scalable:** SCPD has a linear time complexity with respect to the number of edges and is highly scalable. For example, on the MAG-History dataset with 2 million edges, SCPD runs in 29 seconds on a stock laptop with CPU.

- **Attributed:** To the best of our knowledge, SCPD is the first method to incorporate node attributes into change point detection for dynamic graphs. On our original COVID-flight dataset, SCPD leverages the country code of airports (nodes) to identify traffic disturbances due to flight restrictions specific to countries such as China and US.

5.2 Related Work

In this section, we review methods for change point and event detection. We compare SCPD to other approaches in Table 5.1. Note that current methods focus on graph structural anomalies, while SCPD is the first method to incorporate node attributes and satisfies all the desired properties.

Event Detection. Idé and Kashima [97] use the principal eigenvector of the adjacency matrix to represent the graph at each snapshot (called *activity vector*). Koutra et al. [111] formulated dynamic graphs as high-order tensors and proposed to use the PARAFAC decomposition [20, 78] to obtain vector representations for anomaly scoring. SPOTLIGHT [54] was proposed to spot anomalous graphs containing the sudden appearance or disappearance of large dense subgraphs.

Change Point Detection. Wang et al. [207] modeled network evolution as a first-order Markov process and used MCMC sampling to design the EdgeMonitoring method. The LAD method presented in Chapter 4 is another very related work. Both LAD and SCPD utilize spectral information from the Laplacian. However, computing Singular Value Decomposition (SVD) limits LAD to small graphs while SCPD can scale to millions of nodes and edges.

Network Density of States. Dong et al. [45] borrowed tools from condensed matter physics and added adaptations such as motif filtering to design an efficient approximation method for spectral density in large networks. Huang et al. [90] proposed a graph kernel that combines the local and global density of states of the normalized adjacency matrix for the graph classification task. ADOGE [170] is an embedding method for exploratory graph analysis and graph classification on static graphs. To the best of our knowledge, our proposed SCPD is the first method to model spectral density for dynamic graphs.

5.3 Problem Setting

We consider an undirected, weighted, dynamic graph $\mathbf{G}$ with node attributes (optional), as a sequence of graph snapshots, $\{\mathcal{G}_t\}_{t=1}^{T}$. Each $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t, \mathbf{X}_t)$ represents the graph at time $t \in [1 \dots T]$, where $\mathcal{V}_t$, $\mathcal{E}_t$ are the set of nodes and edges respectively, and $\mathbf{X}_t \in \mathbb{R}^{|\mathcal{V}_t| \times N_a}$ is the attribute matrix, where N_a is the number of attributes. An edge $(i, j, w) \in \mathcal{E}_t$ connects node i and node j at time t with weight w. We use $\mathbf{A}_t \in \mathbb{R}^{|\mathcal{V}_t| \times |\mathcal{V}_t|}$ to denote the adjacency matrix of $\mathcal{G}_t$.

Change Point Detection. The goal of change point detection is to identify anomalous time steps in a dynamic graph, i.e. snapshots with *graph structures* that significantly deviate from the normal behavior. This requires an anomaly score function measuring the graph structural differences between the current snapshot and previously observed behaviors. In this work, we examine both *events*, one-time change to the graph structure, and *change points*, permanent alterations on the graph generative process.

Attribute Change Point Detection. To the best of our knowledge, we are also the first work to incorporate node attributes in change point detection. In addition to detecting change points in the graph structure, *attribute* change point detection aims to identify time steps in which *the alignment* between node attributes and graph structure deviates significantly from the norm. For example, in a network with communities, if the distribution of an attribute conditioned on the community drastically changes, we say that an attribute change point has happened.

5.4 Methodology

To detect anomalous snapshots, we embed each graph snapshot into a low-dimensional embedding called the signature vector based on the spectral density. Then, the normal behavior of the graph in the past is summarized into a vector. Lastly, we compare the

Algorithm 1 SCPD

Input: $\mathbb{G} = \{\mathcal{V}_t, \mathcal{E}_t, \mathbf{X}_t\}_{t=1}^{T}$ & attribute a (optional)
parameter: w_s, w_l: short & long sliding window sizes, N_z, N_m: number of probe vectors & Chebyshev moments
Output: anomaly scores Z^* (general) & Z_a^* (attribute)

/* Compute Density of States */
foreach *graph snapshot* $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t, \mathbf{X}_t) \in \mathbb{G}$ **do**
  Compute the symmetric normalized Laplacian $\mathbf{L}_{sym}$ $\boldsymbol{\sigma}_t \leftarrow$ KPM_DOS($\mathbf{L}_{sym}$, N_z, N_m)
  if *attribute* a *is given*
    $\boldsymbol{\sigma}_{t,a} \leftarrow$ GQL_LDOS($\mathbf{L}_{sym}$, $[\boldsymbol{X}_t]_{:,a}$, N_z, N_m)
  L2 normalize both $\boldsymbol{\sigma}_t$ and $\boldsymbol{\sigma}_{t,a}$
/* Compute Anomaly Scores */
foreach *time step* t **do**
  $\tilde{\sigma}_t^{w_s}$ and $\tilde{\sigma}_t^{w_l} \leftarrow$ left singular vector of context $\mathbf{C}_t^{w_s}$ and $\mathbf{C}_t^{w_l}$, where

$$\mathbf{C}_t^{w} = \begin{pmatrix} | & | & & | \\ \boldsymbol{\sigma}_{t-w-1} & \boldsymbol{\sigma}_{t-w-2} & \cdots & \boldsymbol{\sigma}_{t-1} \\ | & | & & | \end{pmatrix} \in \mathbb{R}^{k \times w}$$

  Obtain Z^* score for $\boldsymbol{\sigma}_t$: $Z = \max(1 - \boldsymbol{\sigma}_t{}^\top \tilde{\sigma}_t^{w_s}, 1 - \boldsymbol{\sigma}_t{}^\top \tilde{\sigma}_t^{w_l})$ **if** *d is given*
  $\tilde{\sigma}_{t,a}^{w_s}$ and $\tilde{\sigma}_{t,a}^{w_l} \leftarrow$ left singular vector of context $\mathbf{C}_{t,a}^{w_s}$ and $\mathbf{C}_{t,a}^{w_l}$ Obtain Z_a^* score for $\boldsymbol{\sigma}_{t,a}$
Return Z^*, Z_a^*

signature from the current step with that of the past behavior and derive an anomaly score. For more details on SCPD see Algorithm 1.

5.4.1 Designing Signature Vector

Identifying change points requires the comparison between multiple graph snapshots. In general, it is difficult to compare graphs directly, as shown in the graph isomorphism problem [208, 214]. Therefore, we want to embed each graph snapshot into a low dimensional vector, called the *signature vector*, and facilitate comparisons between vectors rather than graphs. In this work, we choose the (global) density of states (DOS) of the Laplacian matrix as the signature vector, since it has the following desirable properties: 1.)*scalable*, DOS can scale to graphs with millions of nodes and edges, 2.)*independent of graph size*, DOS produces a fixed sized embedding independent of size of the graph, 3.) *incorporates*

attributes, the local DOS models the alignment between node attributes and eigenvectors thus can be used to model attribute change points.

We use DOS to approximate the distribution of the Laplacian eigenvalues. The Laplacian eigenvalues capture many graph structures and properties [188] and have shown strong empirical performance for anomaly detection [93]. For example, the number of zero eigenvalues of the Laplacian matrix is equal to the number of connected components of the graph [200], and the eigenvectors of the Laplacian matrix provide an effective way to represent a graph in a 2D plane [72]. In addition, the eigenvalues of the Laplacian and their multiplicity reflect the geometry of many fundamental graphs, such as complete graphs, star graphs, and path graphs. However, computing all Laplacian eigenvalues of a graph requires $O(|\mathcal{V}| \cdot |\mathcal{E}|)$, which is only practical for small graphs, while DOS can be computed for graphs with millions of nodes. Later in this section, we show how to compute DOS efficiently, and in Section 5.5 and 5.6, we demonstrate that DOS has state-of-the-art performance in change point detection.

5.4.2 Computing Anomaly Score

After computing the signature vectors for each timestamp, now we explain how to detect anomalous snapshots. We assume that when an anomaly arrives, it will be significantly different from recent snapshots. Therefore, we extract the "expected" or "normal" behavior of the dynamic graph from a context window of size w from the past w signature vectors. To obtain unit vectors, $L2$ normalization is performed on the set of the signature vectors $\boldsymbol{\sigma}_{t-w-1}, \ldots, \boldsymbol{\sigma}_{t-1}$. Then, we stack the normalized vectors to form the context matrix $\mathbf{C}_t^w \in \mathbb{R}^{k \times w}$ of time t, where k is the length of the signature vector. We compute the left singular vector of $\mathbf{C}_t^w$ to be the summarized normal behavior vector $\tilde{\sigma}_t^w$ (which can be seen as a weighted average over the context window). A smaller context window can detect more sudden or abrupt changes, while a longer window can model gradual and continuous changes. Therefore, we use a short window with size w_s and a long window with size w_l to detect both events and change points.

Now we can compute the anomaly score at time t as

$$Z_t = 1 - \frac{\boldsymbol{\sigma}_t^\top \tilde{\sigma}_t^w}{\|\boldsymbol{\sigma}_t\|_2 \|\tilde{\sigma}_t^w\|_2} = 1 - \boldsymbol{\sigma}_t^\top \tilde{\sigma}_t^w = 1 - \cos\theta$$

where $\cos\theta$ is the cosine similarity between the current signature vector $\boldsymbol{\sigma}_t$ and the normal behavior vector $\tilde{\sigma}_t^w$. In this way, $Z \in [0, 1]$ and when Z is closer to 1, the current snapshot is significantly different from the normal behavior, thus more likely to be an anomaly. The Z scores from windows of size w_s and w_l are then aggregated by the max operation. To emphasize the increase in anomaly score, we compute the difference in anomaly score with the previous step using $Z_t^* = \min(Z_t - Z_{t-1}, 0)$. Finally, the points with the largest Z^* are selected as anomalies. We show the Z^* score in all figures in this work.

5.4.3 Approximating Spectral Density

For clarity, we drop the t subscript in this section. The Laplacian matrix $\mathbf{L} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ is defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$ where $\mathbf{D} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$, $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ are the diagonal degree matrix and the adjacency matrix. In this work, we use the symmetric normalized Laplacian $\mathbf{L}_{sym}$, defined as,

$$\mathbf{L}_{sym} = \mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$$

to present the graph at each snapshot. Consider the eigendecomposition of $\mathbf{L}_{sym} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$ where $\mathbf{\Lambda} = diag(\lambda_1, \ldots, \lambda_{|\mathcal{V}|})$ and $\mathbf{Q} = [\boldsymbol{q}_1, \ldots, \boldsymbol{q}_{|\mathcal{V}|}]$ is an orthogonal matrix. We can now define the Density of States or the spectral density as,

Definition 1 (Density of States (DOS))**.** *The global density of states or spectral density induced by* $\mathbf{L}_{sym}$ *is:*

$$\mu(\lambda) = \frac{1}{|\mathcal{V}|}\sum_{i=1}^{|\mathcal{V}|}\delta(\lambda - \lambda_i) \tag{5.1}$$

where δ *is the Dirac delta function and* λ_i *is the* i*-th eigenvalue.*

Intuitively, $\mu(\lambda)$ measures the portion of eigenvalues that are equal to λ. In practice, we discretize the range of λ into equal sized intervals and approximate how many λs

fall within each interval. Therefore, across all intervals, the shape of the distribution of eigenvalues are approximated. We use the Kernel Polynomial Method (KPM) [45] to approximate the density function through a finite number polynomial expansion in the dual basis of the Chebyshev basis, and the spectrum is adjusted to be in $[-1, 1]$ for numerical stability.

To incorporate attributes, we also consider the Local Density of States:

Definition 2 (Local Density of States (LDOS))**.** *For any given input vector $\boldsymbol{v} \in \mathbb{R}^N$, the local density of states is:*

$$\mu(\lambda; \boldsymbol{v}) = \sum_{i=1}^{|\mathcal{V}|} |\boldsymbol{v}^T \boldsymbol{q}_i|^2 \delta(\lambda - \lambda_i) \tag{5.2}$$

where $\boldsymbol{v}$ is an input vector, and λ_i and $\boldsymbol{q}_i$ are the i-th eigenvalue and eigenvector, respectively.

The term $\boldsymbol{v}^T \boldsymbol{q}_i$ acts as a weight on the ith bin of the spectral histogram. To incorporate node attributes, we set $\boldsymbol{v} = \boldsymbol{x}$ where $\boldsymbol{x} \in \mathbb{R}^{|\mathcal{V}|}$ is an attribute vector. This can be interpreted as the *alignment* between the node attribute vector and the graph structure of the group of nodes with such attribute. As the alignment is measured in each eigenvalue interval (similarly to DOS), we obtain a LDOS embedding of size k for each attribute and each possible category. By tracking this embedding over time, the anomalous evolution specific to the given attribute is captured. Here, categorical attributes are one-hot encoded, and numerical attributes are normalized by the sum. The Gauss Quadrature and Lanczos (GQL) [45] method to approximate the LDOS with attribute vectors.

5.4.4 Computational Complexity

For unattributed dynamic graphs, SCPD has the complexity of $O(N_z \cdot N_m \cdot |\mathcal{E}|)$ for a given snapshot with $|\mathcal{E}|$ edges. N_z and N_m are hyperparameters in the KPM computation representing the number of probe vectors and Chebychev moments, respectively. For all experiments, we set $N_z = 100$, $N_m = 20$. We also use $k = 50$ equal-sized bins to divide up the range of eigenvalues.

For attributed dynamic graphs, we use the GQL method to compute LDOS for attribute change point detection. GQL method performs the eigendecomposition of a tridiagonal matrix with $O(|\mathcal{V}|^2)$ worst-case complexity. Note that in practice, such computation is very fast [170]. Therefore, for a given attribute on a dynamic graph, SCPD's time complexity is $O(\eta \cdot |\mathcal{E}| + |\mathcal{V}|^2)$ for a given snapshot. In practice, SCPD is very fast, only costing 5 seconds to run on the COVID flight network with close to 1 million edges and 5 node attributes with an AMD Ryzen 5 1600 Processor and 16GB memory.

5.5 Synthetic Experiments

In this section, we conduct experiments with the Stochastic Block Model (SBM) [83] and the Barabási–Albert (BA) model [5] as synthetic graph generators and plant 7 ground truth anomalies for all experiments. We report the Hits@n metric the same as in [93] and the execution time over 5 trials. We discuss our considered baselines and contenders in Section 5.5.1, evaluation setting in Section 5.5.2, and planted anomalies in Section 5.5.3.

5.5.1 Contenders and Baselines

We compare SCPD with the current state-of-the-art baselines. The same short-term and long-term window sizes ($s = 5$ and $l = 10$) are used if applicable. For methods with stochasticity, we report the average result over 5 runs.

- **SPOTLIGHT [54] with RRCF [69]**: SPOTLIGHT is a randomized sketching-based approach to detect anomalous graphs containing sudden appearance or disappearance of large dense subgraphs. We implemented SPOTLIGHT from scratch in Python and then used the recommended anomaly detection method Robust Random Cut Forest (RRCF) [69] for the SPOTLIGHT embeddings. Following [54], we set $p = q = 0.2$ and $k = 50$ sketch dimensions.

- **SPOTLIGHT [54] with a sum, or SPOTLIGHTs**: to achieve a more competitive performance with SPOTLIGHT for change point detection, we introduce a simple anomaly detection pipeline to use with the computed SPOTLIGHT embeddings. We find that simply summing over all SPOTLIGHT dimensions in each snapshot is a good proxy for the anomaly score, and we use the difference between the sum score of t and $(t-1)$ as the final anomaly score.

- **LAD [93]**: Laplacian Anomaly Detection (LAD) utilizes all the eigenvalues of the Laplacian matrix as the embedding vector for each snapshot. For LAD, we use the symmetric normalized Laplacian $\mathbf{L}_{sym}$ instead of unnormalized Laplacian $\mathbf{L}_t$ because $\mathbf{L}_{sym}$ improved LAD performance across both BA and SBM datasets. Using the recommendation from [93], we compute all the eigenvalues in the Laplacian spectrum to achieve the best performance for LAD. Due to the high cost of SVD, we limit the computational time to less than 5 hrs, and if the computation is not finished within such time limit, we report N/A as it is considered not scalable.

- **EdgeMonitoring [207]**: EdgeMonitoring uses joint edge probabilities as the feature vector and models network evolution as a first-order Markov process. We use the official MATLAB implementation kindly provided by the authors of EdgeMonitoring and sample 250 edges in 25 equal-sized groups as in [207]. The detected anomalies are based on the Euclidean Distance.

5.5.2 Performance Evaluation

We conduct experiments with the Stochastic Block Model (SBM) and the Barabási–Albert (BA) model as synthetic graph generators and plant ground truth anomalies for all experiments. Similar to [93], we use the Hits@n metric, which reports the number of correctly detected anomalies out of the top n steps with the highest anomaly scores. For synthetic experiments, for uniformity, we plant 7 anomalies and report Hits@7 (which penalizes both false positive and false negative anomaly alarms). We also report the execution time

in seconds on a desktop with AMD Ryzen 5 1600 CPU and 16 GB memory. We use the ground truth labels from the generation process for evaluation.

5.5.3 Planted Anomaly Details

In this section, we describe the details of the planted anomalies and the random graph generators used in Section 5.5. Table 5.2 a). and b). describes the anomalies in the SBM hybrid experiment and SBM attribute experiment, respectively. Table 5.3 provides details for the anomalies in the SBM evolving size experiment and the BA experiment.

SBM [83] is a widely used graph generation model for community structures. The key parameters of the SBM model are: 1). the partitioning of communities, 2). the intra-community connectivity p_{in}, and 3). the cross-community connectivity p_{out}. For simplicity, we assume equal-sized communities and instead focus on changing the number of communities N_c. p_{in} and p_{out} determine the probability of an edge existing between nodes of the same community and different communities, respectively. They also control the sparsity of the dynamic graph.

5.5.4 SBM Hybrid Experiment

We follow the Hybrid setting in [93]. SBM [83] is used to generate equal-sized communities with p_{in} being the intra-community connectivity and p_{out} being the cross-community connectivity. Change points are the merging or splitting of communities in the dynamic graph, and events are one-time boosts in cross-community connectivity p_{out}. Figure 5.1 shows that SCPD perfectly identifies all the events and change points on a dynamic SBM graph. We also visualize the signature vectors (the computed DOS or distribution of eigenvalues) as a heatmap. The events (time points 16,61,91,136) correspond to an energetic burst in the signature vector. And the change points correspond to the shifts in the distribution of Laplacian eigenvalues. Interestingly, the width of the distribution seems to correlate with the number of communities N_c.

Table 5.2: Experiment Setting: the changes in the generative model in
(a) the setting for the SBM hybrid experiment where both events and change points occur in a SBM model, N_c is the number of equal-sized communities, p_{in} is the intra-community connectivity and p_{out} is the cross-community connectivity.
(b) the setting for the SBM attribute experiment where change points in the node attribute and graph structure of a SBM model occur

SBM Change Points Details				
Time	Type	N_c	p_{in}	p_{out}
0	start point	4	0.030	0.005
16	event	4	0.030	**0.015**
31	change point	**10**	0.030	0.005
61	event	10	0.030	**0.015**
76	change point	**2**	0.030	0.005
91	event	2	0.030	**0.015**
106	change point	**4**	0.030	0.005
136	event	4	0.030	**0.015**

(a) anomalies in the SBM hybrid experiment.

SBM Node Attribute and Structural Changes			
Time	Type	N_c	Node Attributes
0	start point	4	homogeneous
16	change point	4	**heterogeneous**
31	change point	**10**	heterogeneous
61	change point	10	**homogeneous**
76	change point	**2**	homogeneous
91	change point	2	**heterogeneous**
106	change point	**4**	heterogeneous
136	change point	4	**homogeneous**

(b) anomalies in the SBM attribute experiment

5.5.5 SBM Attribute Experiment

We want to demonstrate SCPD's ability to detect anomalous evolution of the node attributes in a dynamic graph. A SBM model is used to construct communities for nodes while each node has a binary attribute. The attributes within a community can be either *homogeneous* or *heterogeneous*. In a homogeneous community, all nodes have the same attribute, while half of all communities have label one and the other half have label two. In a heterogeneous community, each node has 0.5 probability of being either one or two, and the node attribute is no longer dependent on community structure. The change points are time points where the node attributes change to *homogeneous* or *heterogeneous*. SCPD is able to recover all change points (16,61,91, and 136) related to node attributes and detect both the change from homogeneous communities to heterogeneous ones as well as the reverse.

Table 5.3: Experiment Setting: the changes in the generative model in
(a) setting for the SBM hybrid experiment where both events and change points occur in a SBM model, N_c is the number of equal-sized communities, p_{in} is the intra-community connectivity and p_{out} is the cross-community connectivity.
(b) setting for the BA experiment where the parameter m is the number of edges to attach from a new node to existing nodes. Higher m value = higher color intensity.

SBM Anomaly Details			
Time	Community Sizes	p_{in}	p_{out}
0	300,300	0.030	0.005
16	300,300,**300**	0.030	0.005
31	300,300,**300,300**	0.030	0.005
61	300,300,**150,150,150,150**	0.030	0.005
76	300,300,**300,300**	0.030	0.005
91	**150,150,150,150**,300,300	0.030	0.005
106	**300,300**,300,300	0.030	0.005
136	300,300,300,300	0.030	**0.015**

(a) anomalies in SBM Evolving Size Experiment

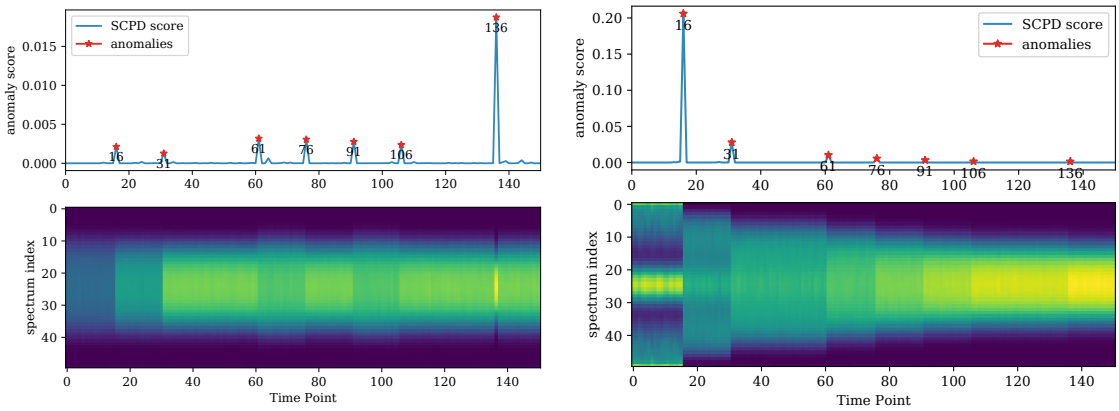
BA Anomaly Details		
Time	Type	m
0	start point	1
16	change point	2
31	change point	3
61	change point	4
76	change point	5
91	change point	6
106	change point	7
136	change point	8

(b) anomalies in the BA experiment.

Interestingly, by just using LDOS embedding, SCPD also captures two out of three structural change points when the number of communities changes. This is because, in Equation 5.2, the $\boldsymbol{v}^T\boldsymbol{q}_i$ term measures the alignment between the input attribute vector and the eigenvector corresponding to ith eigenvalue of the Laplacian. This term would change either due to a change in $\boldsymbol{v}$ or $\boldsymbol{q}_i$ or both. In this case, the change in community structure also caused a change in the eigenvectors of the Laplacian. Note that as none of the alternative methods take into account the node attributes, SCPD is the only method that can detect the anomalous evolution of node attributes.

5.5.6 SBM Evolving Size Experiment

We examine SCPD's ability to adapt to the evolving size of a dynamic graph (with a SBM as the graph generator). Initially, there are two communities with 300 nodes each. Later on, additional nodes are added and forming a total of 4 communities. Some change points



(a) SBM evolving size experiment

(b) The BA experiment

Figure 5.2: a). SCPD output for the SBM evolving size experiment
b). SCPD output for the BA experiment.

involve only nodes from the initial step, while some involve only newly added nodes. Figure 5.2a shows that SCPD is able to adapt to dynamic graphs with evolving sizes and correctly predicts anomalies in the global graph structure. Only SCPD and LAD are able to correctly detect all anomalies, while SPOTLIGHT and EdgeMonitoring can only detect changes local to the initial set of nodes. This shows that SCPD can effectively adapt to the evolving size of dynamic graphs.

5.5.7 BA Experiment

We evaluate SCPD performance in a different graph distribution, the BA model. In this experiment, the change points correspond to the densification of the network (parameter m, increased number of edges attached from a new node to an existing node). From Figure 5.2b, we see that SCPD is able to detect all change points in the BA model and the most drastic change in DOS happens when m changes from one to two and the graph becomes connected. This is because the number of zero eigenvalues in the Laplacian matrix corresponds to the number of connected components in the graph; thus, when the graph is connected, the smallest eigenvalue intervals become less energetic.

Table 5.4: SCPD can efficiently operate on large graphs while achieving the state-of-the-art performance. Each dynamic graph has 151 time steps. The results are Hits@7 averaged over 5 trials and the mean and standard deviations are reported. We consider a method not applicable (N/A) if the computation takes longer than 5 days.

Generator	SBM			BA	
Experiment	Hybrid		Evolving Size	Change Point	
Total Edges (millions)	0.8 m	56.9 m	1.0 m	0.6 m	5.5 m
SCPD (ours)	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**
LAD [93]	**1.00 ± 0.00**	N/A	**1.00 ± 0.00**	**1.00 ± 0.00**	N/A
SPOTLIGHT [54]	0.31 ± 0.06	0.57 ± 0.00	0.20 ± 0.07	0.06 ± 0.07	0.11 ± 0.11
SPOTLIGHTs	0.71 ± 0.00	0.71 ± 0.00	0.31 ± 0.06	**1.00 ± 0.00**	**1.00 ± 0.00**
EdgeMonitoring [207]	0.06 ± 0.11	0.00 ± 0.00	0.14 ± 0.00	0.06 ± 0.07	0.17 ± 0.11

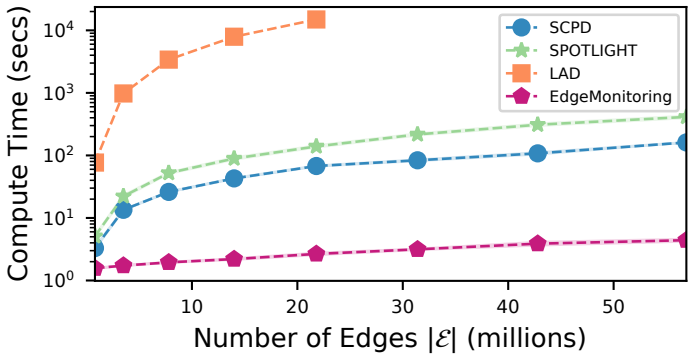


Figure 5.3: Compute time comparison between different methods on the SBM hybrid experiment with varying number of edges.

5.5.8 Summary of Results

Table 5.4 compares the performance of SCPD with state-of-the-art methods on synthetic experiments. The SBM attribute experiment is not included, as only SCPD can incorporate node attributes. The considered baselines include LAD [93], SPOTLIGHT [54] and EdgeMonitoring [207]. The original SPOTLIGHT (with RRCF [69] detector) and our own variant, SPOTLIGHT with sum predictor, called SPOTLIGHTs are both included. Across all experiments, SCPD has the best overall performance. With the default RRCF anomaly detection pipeline, SPOTLIGHT [54] performs poorly on the BA model and middling per-

formance on the SBM hybrid experiment. With the simple sum predictor introduced by us, SPOTLIGHTs is a much closer competitor with a strong performance on the BA model and improved performance on the SBM hybrid experiment. However, SPOTLIGHTs is still not able to detect changes in the evolving size experiment and overall outperformed by SCPD. EdgeMonitoring [207] has low performance in the synthetic experiments due to its dependency on node ordering as well as the assumption that only a small percentage of edges would be resampled in a dynamic graph. The closest competitor to SCPD is LAD [93]. However, computing all the eigenvalues in LAD is prohibitively expensive on large graphs and thus reported as not applicable.

In Figure 5.3, we compare the computational time across different methods in the SBM hybrid experiment. The most expensive is LAD as it has worst-case complexity cubic to the number of nodes, thus having a poor trade-off between performance and efficiency. In contrast, both SCPD and SPOTLIGHT have complexity linear to the number of edges. However, SCPD outperforms SPOTLIGHT across all experiments shown in Table 5.4. Therefore, SCPD has the best trade-off between compute time and performance. Lastly, EdgeMonitoring has sublinear complexity to the number of edges; however, its performance is not ideal. Empirically, we also observe that SCPD is robust to the choice of hyperparameters, including the number of probing vectors N_z, the number of Chebychev moments N_m, and the number of equal-sized bins k in the range of Laplacian eigenvalues.

5.6 Real World Experiments

We empirically evaluate SCPD on two real-world dynamic networks and cross-reference anomalies detected by SCPD with significant events.

5.6.1 MAG History Co-authorship Network

MAG-History is a co-authorship dynamic network extracted from the Microsoft Academic Graph (MAG) [184, 12] by identifying publications which are marked with the

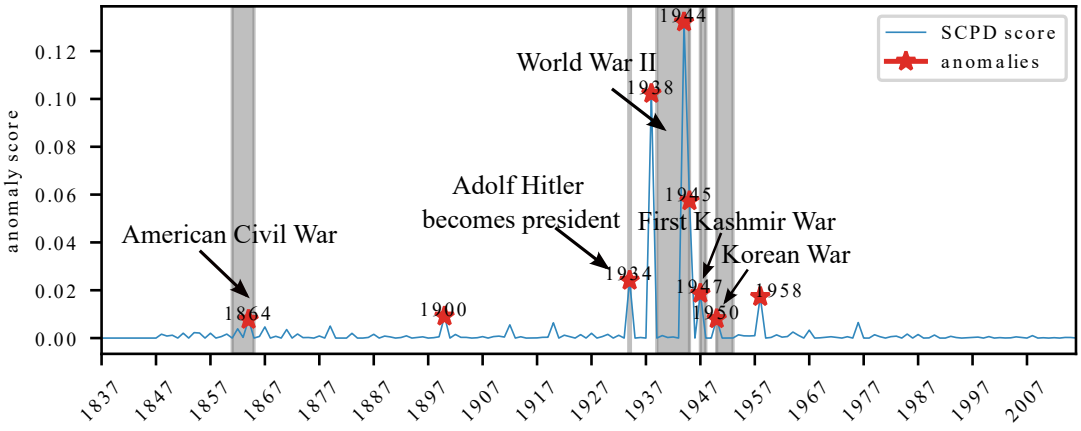


Figure 5.4: SCPD detects significant historical events from the MAG-History dataset.

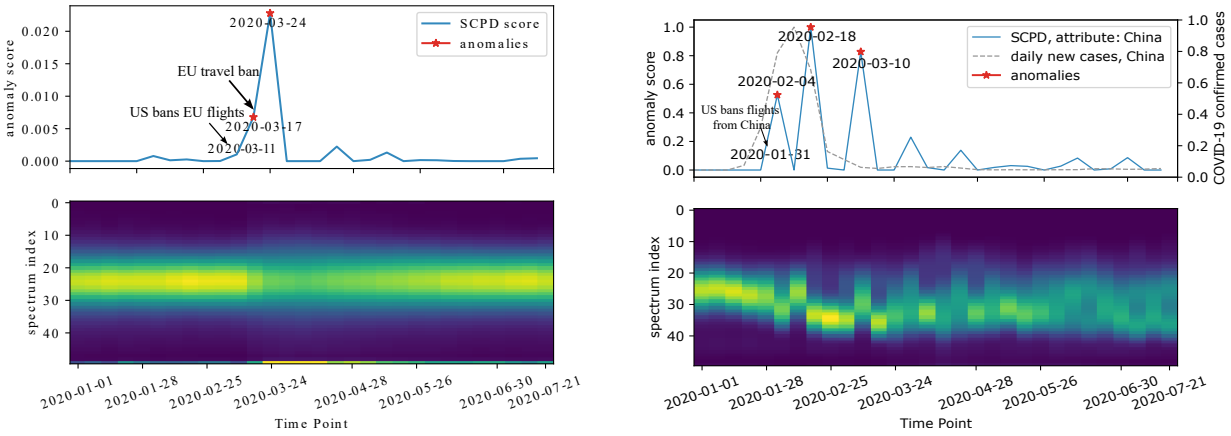
"History" tag. The processed dataset is an undirected dynamic graph from 1837 to 2018. There are 2.8 million projected edges across all time steps and 0.7 million nodes in total. To compute the DOS embedding for this dataset, SCPD only takes 30 seconds.

Figure 5.4 shows the anomalies detected by SCPD. Interestingly, many of the anomalies correspond to important historical events such as the American Civil War (1861-1865), Adolf Hitler's rise to power (1934), the Second World War (1939-1945), the First Kashmir War (1947-1948) and Korean War (1950-1953). The relation between the change in co-authorship graph structure and these historical events can be an interesting direction for future work. In comparison, both variants of SPOTLIGHT miss the Second World War as a top anomaly, while EdgeMonitoring's output is noisy, and many data points share high anomaly scores.

5.6.2 COVID Flight Network

The COVID flight network [†] [173, 145] is a dynamic air traffic network during the COVID-19 pandemic. The nodes are airports, and each edge is an undirected timestamped tracked flight with the frequency as edge weight. We examine the period from 01-01-2020 to 07-

[†] https://zenodo.org/record/3974209/#.Yf62HepKguU



(a) Graph structural anomalies.

(b) Node attribute anomalies.

Figure 5.5: a). SCPD detects the week of 03.17 and 03.24 as structural anomalies in the global flight network in 2020. On 03.17, the European Union closed its borders to travelers, thus causing widespread disruption in international flights.
b). SCPD detects the closure of flight routes to China due to COVID interventions at the beginning of Feb 2020. The anomaly score and case numbers are normalized to $[0, 1]$

27-2020. We use a full week as the duration of each snapshot to reduce the noise and variability from daily flights. Figure 5.5a shows the graph structural anomalies detected by SCPD using the DOS embeddings as signature vectors. The two weeks with the highest anomaly scores are 03-17-2020 and 03-24-2020. On 03-17, the European Union adopted a 30-day ban on non-essential travel to at least 26 European countries from the rest of the world (see here). On 03-11, the US President banned travel from 26 European countries. These events are detected by SCPD as travel bans severely disrupt the international flight network. In comparison, SPOTLIGHT detects the week of 02-11 corresponding to flight restrictions in China, while EdgeMonitoring also detects mid-March as anomalies.

Figure 5.5b shows SCPD's detected anomalies when the node attribute is set to be an indicator vector for which nodes are Chinese airports. The detected anomalies lie mainly in February and early March because the COVID outbreak was first detected in China in January 2020. On 01-31-2020, the Trump administration suspended entry into the United States by any foreign nationals who had traveled to China in the past 14 days (see here).

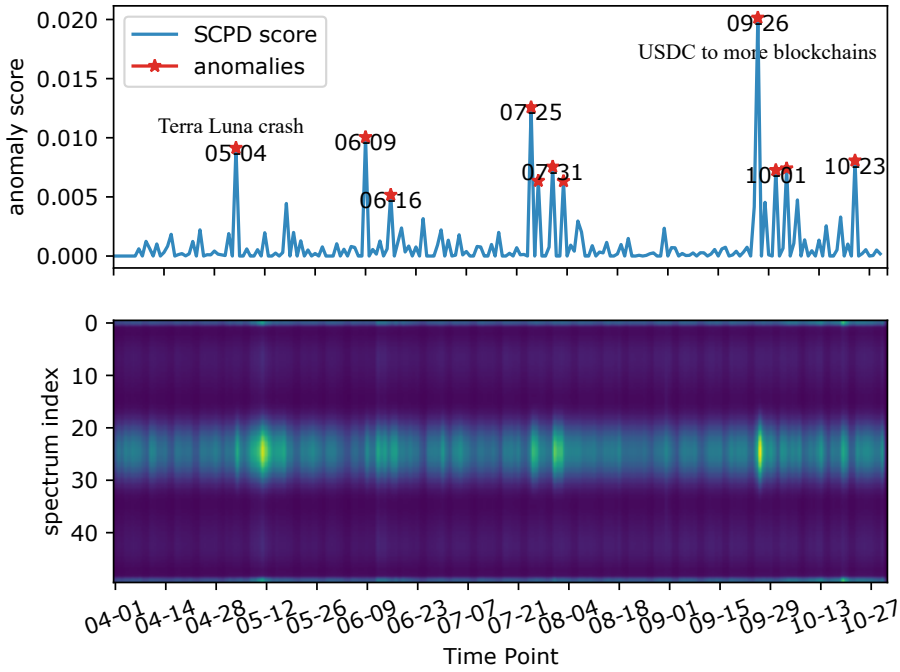


Figure 5.6: SCPD detects the Lunar coin crash on the Stablecoin transaction datset.

Therefore, the anomaly observed by SCPD on the week of 02-04-2020 is likely the directed result of the imposed travel restriction. Note that Figure 5.5b shows that the peak of new daily cases in China [‡] corresponds to a peak in anomaly score, likely because of reduced domestic and international flights at that time. For Canada, SCPD also detects the implementation of travel restrictions. Therefore, SCPD captures both the structural anomalies in the flight network as well as anomalies specific to the set of nodes with the same attribute.

5.6.3 Stablecoin Transaction Network

The Stablecoin transaction dataset [§] [177] tracks the transaction network of six stablecoin networks. The nodes are contact addresses and the edges are transactions. The dataset

[‡] https://www.worldometers.info/coronavirus/country/china/
[§] https://www.chartalist.org/eth/StablecoinAnalysis.html

spans from April first 2022 to November first 2022, and the most notable anomaly or change point is the Terra Luna crash when it lost its $1 USD fixed price value in early May 2022. Note that Stablecoins are special tokens that are meant to maintain a fixed price value, such as the $1 USD per token price mentioned above. Therefore, the Terra Luna crash has significant impact on the entire transaction network. Figure 5.6 shows the detected change points by SCPD. Most notably, SCPD is able to detect two of the most significant events in the stablecoin network in the duration of the dataset. Notable, on May 4th, SCPD shows the first significant jump in anomaly score, which corresponds to the Terra Luna Crash on May 7th. Signaling the anomaly 3 days early is a highly desirable behavior in transaction networks. In addition, the peak with the highest anomaly score occurred on September 26th when USDC came to five additional blockchain ecosystems.

5.7 Conclusion

In this chapter, we proposed a novel change point detection method, SCPD, to detect anomalous changes in the graph structure as well as node attributes in a dynamic graph. SCPD approximates the distribution of Laplacian eigenvalues as an embedding for the graph structure and Local DOS embeddings to measure the alignment between node attributes and the eigenvectors of the Laplacian at different frequency intervals. In synthetic experiments, SCPD achieves state-of-the-art performance while running efficiently on graphs with millions of edges. On three real-world datasets, SCPD is able to capture structural and attribute change points corresponding to significant events.

Chapter 6

Laplacian Change Point Detection for Multi-view Dynamic Graphs

In Chapter 4 and 5, we proposed novel methods for change point detection on single-view dynamic graphs. However, real-world networks may contain a diverse range of information sources and thus should be modeled as multi-view dynamic graphs. In this chapter, we investigate if the change point detection problem in dynamic graphs can benefit from including more views and how to aggregate the most useful information from each view for anomaly detection.

To this end, we propose MultiLAD, a simple and effective generalization of LAD to multi-view graphs. MultiLAD provides the first change point detection method for multi-view dynamic graphs. It aggregates the singular values of the normalized graph Laplacian from different views through the scalar power mean operation.

Through extensive synthetic experiments, we show that i) MultiLAD are accurate and outperforms state-of-the-art baselines and their multi-view extensions by a large margin, ii) MultiLAD 's advantage over contenders significantly increases when additional views are available, and iii) MultiLAD is highly robust to noise from individual views. In two real-world multi-view traffic networks, we demonstrate that MultiLAD identifies meaningful real-world events as top anomalies. Notably, in the New York City 2020 dataset,

MultiLAD accurately detects important dates corresponding to the implementation of government COVID-19 interventions that impacted the population mobility. This chapter is based on our publication at the TKDD journal [91].

Reproducibility: code for this chapter is available on Github [*].

6.1 Introduction

In practice, a number of networks can describe the same underlying relations. Multi-view (a.k.a. multiplex) networks are a subset of multi-layer networks that describe relations between the same set of entities from different information sources. Each source can be modeled as an individual view. The interactions between the entities in each view often evolve over time thus forming multi-view dynamic networks. These networks arise naturally in numerous domains such as traffic/transportation networks [60, 54], mobile phone networks [109], friendship networks [49], social media networks [204] and citation networks [193]. Recent work has shown promising improvements by leveraging the additional information from multi-view data in tasks such as community detection [151, 138], anomalous node detection [179] and event forecasting [89]. To the best of our knowledge, existing methods studying change point detection only focused on single-view dynamic networks [93, 207, 97, 3, 155]. Therefore, we propose MultiLAD as the first change point detection method for multi-view dynamic networks.

To better show the motivation for MultiLAD, let us consider a real-world application. Given trip records of different taxi companies and for-hire vehicles, can we identify important occasions such as holidays, events, or unusual weather conditions that disrupt the overall traffic network? In general, there are two main motivations for leveraging multi-view information for change point detection in dynamic networks. First, data from additional sources may have different characteristics and values [151] thus contributing

[*] https://github.com/shenyangHuang/multiLAD

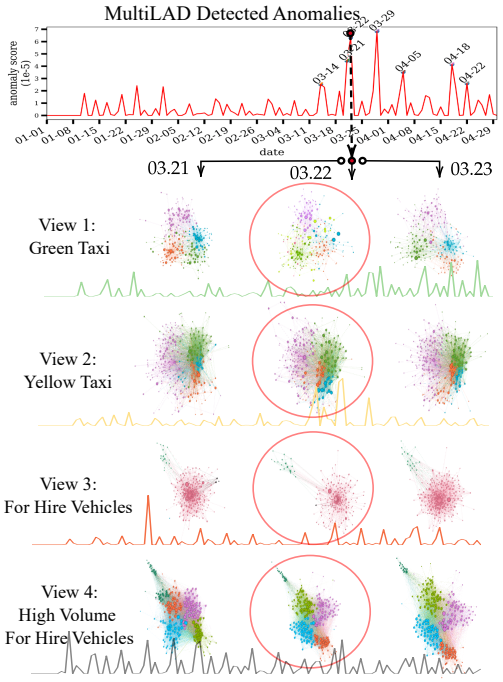


Figure 6.1: Our proposed MultiLAD accurately detects March 22nd 2020 as an anomalous day in the New York City Transit network. This day is the first day of NYS on Pause Program where all non-essential workers must stay home [105]. Per each view, we visualize the graph structure on the detected change point as well as the day before and after this point. The anomaly scores from the single-view method LAD are also reported for each view to demonstrate the challenge of combining this information when derived independently. On top, we show that MultiLAD's anomaly score (which considers these four views jointly) can effectively reason across views.

unique and complementary information. In this case, records from taxi companies and for-hire vehicle services (such as Uber and Lyft) should model the same traffic network but from potentially different age groups and demographics. Second, individual sources can have a high degree of noise and contain disruptive information due to the data collection process. Figure 6.1 shows the anomalous event on March 22nd 2020 in the New York City Transit network detected by our proposed method. This day signals the beginning of the work-from-home requirement for all non-essential workers. To better illustrate the graph structure, the nodes are colored by their detected community assignment and their sizes correspond to PageRank [148]. Here, we have four views which are explained in

detail in Section 6.6.9. Given these four views with different graph structures and evolution, our method effectively reasons across views to find the important anomalies in the underlying traffic network. In contrast, it is often difficult or even infeasible to select one view and apply a single-view change point detection method such as LAD (LAD's anomaly scores on each individual view are reported in Figure 6.1). As seen in the figure, each view can exhibit a different set of anomalies, leading to significant differences among anomaly scores. This disagreement problem is further exacerbated when more views are present. Therefore, naive aggregation strategies on anomaly scores (such as max or mean) are insufficient, while MultiLAD aggregates view-specific features in a meaningful way, preserving the essential information for anomaly detection. After view aggregation, MultiLAD utilizes the same anomaly detection procedures as LAD thus MultiLAD can be seen as a multi-view generalization of LAD.

6.2 Multi-view Change Point Detection

based on the above definition, let $\mathcal{G}$ be a multi-view dynamic graph and let $\mathbf{H}_t$ be the underlying graph generative model at time t. $\mathbf{H}_t$ is not directly observed but is assumed to control the graph behavior across all views. The goal of multi-view change point detection is then to find time points after which $\mathbf{H}_t$ significantly differs from the previous steps. More precisely, we want to find a set $S \subseteq \{1, ..., T\}$ such that for each $t \in S$, $\cdots \simeq \mathbf{H}_{t-2} \simeq \mathbf{H}_{t-1} \not\simeq \mathbf{H}_t \simeq \mathbf{H}_{t+1} \simeq \cdots$. Note that when there is only 1 view, this becomes single-view change point detection as defined in [207, 97].

Multi-view Event Detection. Here, we also consider events similar to [97]. Let $\mathbb{G}$ be a multi-view dynamic graph and let $\mathbf{H}_t$ be the underlying graph generative model at time t. The goal of event detection is to find a set of time points $S \subseteq \{1, ..., T\}$ such that for each $t \in S$, $\mathbf{H}_{t-1} \not\simeq \mathbf{H}_t$ and $\mathbf{H}_{t-1} \simeq \mathbf{H}_{t+1}$. Events are one-time sudden changes in graph structure where the $\mathbf{H}_{t+1}$ reverts to normal after the event at time t.

6.3 Related Work

Here we discuss related work of machine learning on multi-view graphs. Related literature on change point detection can be found in Chapter 3.2.1.

Multi-view Anomaly Detection. Most multi-view anomaly detection methods follow a two-step procedure: 1). extract view-specific representations and 2). aggregate such representations to identify anomalous data. While prior work [194, 179, 70, 186] focused on the anomalous node (or instance) detection task, we tackle the change point detection problem in the multi-view *dynamic* graph setting. In other words, instead of detecting anomalous nodes, we spot anomalous timestamps when the structure of the entire graph changes.

We draw on the same idea used in node-level anomaly detection methods which assumes that abnormal events would create a disturbance of regularities across all views. By mining such irregular patterns, one can achieve stronger anomaly detection results than with just a single view. In this work, we use this idea to identify anomalous graph snapshots across different views to detect significant changes in the overall graph generative process for all views.

Multi-view Clustering. The goal of graph clustering is to obtain groups of nodes that are similar with regards to some structural or node attribute information [151, 192]. Graph clustering in the multi-view setting is a similar task to multi-view change point detection in which one needs to combine information from different sources on the same nodes to extract some underlying structure. However, multi-view graph clustering focuses on static graphs while in this work we consider dynamic graphs.

Mercado *et al.* [138] propose to take the matrix power means of Laplacians (PML) to extend the spectral clustering algorithm to multilayer graphs. Inspired by the effectiveness of power mean operations in multi-view clustering, we propose to utilize the scalar power mean operation to merge information from different views for change point

detection. Sohn *et al.* [186] proposed a Bayesian multilayer stochastic block modeling framework for multilayer clustering. They also extend their method to detect community changes in multilayer temporal graphs. However, their approach is limited to networks with block structures. In comparison, MultiLAD is applicable to any graph including those with no known community structures (such as the Barabási-Albert model).

6.4 Methodology

Our proposed Multi-view Laplacian Anomaly Detection (MultiLAD) uses the scalar power mean operation to meaningfully aggregate information from individual views and infers the anomaly score based on an aggregated vector. More specifically, MultiLAD first computes the singular values of the normalized Laplacian matrix from each view as the view-specific feature vector. Then, MultiLAD merges information across different views by aggregating these feature vectors through the scalar power mean operation. In this way, the effect of noise from individual views is lessened while key information about the overall generative process of the multi-view network is preserved. Lastly, MultiLAD compares the aggregated vector with past normal behavior and determines if a particular time step is anomalous or not. The overall procedure for MultiLAD is summarized in Algorithm 2.

6.4.1 Extracting Signature Vectors Per View

Using the unnormalized Laplacian matrix $\mathbf{L}$ can be problematic in the multi-view setting where edge weights from different views may vary significantly in magnitude. Therefore, we use the symmetric normalized Laplacian [200] instead:

$$\mathbf{L}_{sym} = \mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}} \tag{6.1}$$

Algorithm 2 MultiLAD

Input: Multi-view graph $\mathbb{G}$
parameter: Power p, sliding window sizes w_s, w_l, embedding size k
Output: Final anomaly scores Z^*

foreach *multi-view snapshot* $\mathcal{G}_t$ *in the multi-view graph* $\mathbb{G}$ **do**
  foreach *graph snapshot* $\mathbf{G}_{t,r} \in \mathcal{G}_t$ **do**
    Compute $\mathbf{L}_{sym}$ (see Eq. (6.1)) Compute top k singular values $\tilde{\sigma}_{t,r}$ of $\mathbf{L}_{sym}$
  Let $\mathbf{\Sigma}_t = s_p(\tilde{\sigma}_{t,1}, \ldots, \tilde{\sigma}_{t,r})$ Perform L2 normalization on $\mathbf{\Sigma}_t$ Compute left singular vector $\tilde{\Sigma}_t^{w_s}$ of context $\mathbf{C}_t^{w_s} \in \mathbb{R}^{k \times w_s}$ Compute left singular vector $\tilde{\Sigma}_t^{w_l}$ of context $\mathbf{C}_t^{w_l} \in \mathbb{R}^{k \times w_l}$ $Z_t^{w_s} = 1 - \mathbf{\Sigma}_t^\top \tilde{\Sigma}_t^{w_s}$ $Z_t^{w_l} = 1 - \mathbf{\Sigma}_t^\top \tilde{\Sigma}_t^{w_l}$
foreach *time step* t **do**
  $Z^*_{s,t} = max(Z_{w_s,t} - Z_{w_s,t-1}, 0)$ $Z^*_{l,t} = max(Z_{w_l,t} - Z_{w_l,t-1}, 0)$ $Z^*_t = max(Z^*_{s,t}, Z^*_{l,t})$
Return Z^*

In this way, the Laplacian matrix of each view is normalized by node degree while also being symmetric (the eigenvalues are the same as singular values). Note that the singular values λ_i of $\mathbf{L}_{sym}$ are now bounded in $[0, 2]$ (in contrast with the singular values σ_i of the un-normalized Laplacian which was unbounded). We consider the singular values vector $\boldsymbol{\lambda}_{r,t} = [\lambda_1, \ldots, \lambda_n]$ of the normalized Laplacian $\mathbf{L}_{sym}$ as the signature vector for the graph $\mathbf{G}_{t,r}$ of the r-th view at time step t . Therefore, at each time step t, we obtain m signature vectors coming from each view.

6.4.2 Aggregating Per View Signatures

To aggregate the per-view signature vectors for multi-view change point detection, we propose to use the *scalar power mean* operation. The pth order power mean of a set of non-negative real numbers $x_1, \ldots, x_m$ is defined by:

$$s_p(x_1, \ldots, x_m) = \left(\frac{1}{m} \sum_{i=1}^{m} x_i^p \right)^{\frac{1}{p}} \tag{6.2}$$

where $p \in \mathbb{R}$ is a hyper-parameter. We denote the vector of singular values of the Normalized Laplacian matrix $\mathbf{L}_{sym}$ in view r at time t by $\boldsymbol{\lambda}_{r,t}$ as before. We now define the

scalar power mean spectrum Λ_t of a multi-view dynamic graph with m views at time t as:

$$\Lambda_t = s_p(\boldsymbol{\lambda}_{1,t}, \ldots, \boldsymbol{\lambda}_{m,t}) \in \mathbb{R}^n \tag{6.3}$$

where s_p is applied component-wise to the singular value vectors, i.e., $(\Lambda_t)_i = s_p((\boldsymbol{\lambda}_{1,t})_i, \ldots, (\boldsymbol{\lambda}_{m,t})_i)$ for all $0 \leq i \leq n$. In this way, $\Lambda_t \in \mathbb{R}^n$ is aggregated from the set of per-view signature vectors.

Aggregating different views with the power mean operation is a natural choice as it simply smooths the spectrums across all views thus reflecting the geometry of the overall graph generative model. In addition, the choice of order p gives the flexibility to emphasize different frequencies in the spectrum. In this work, we set $p = -10$ for all experiments which allows the smaller eigenvalues to be emphasized. This is motivated by the fact that the smallest eigenvalues of the Laplacian correspond to the lower frequency in the graph which is often considered to be more useful in practice. For example, the number of zero eigenvalues corresponds to the number of connected components in the graph [200]. The power mean operation is also found to be a powerful tool in the multi-view spectral clustering task [138]. Empirically, we also observe that $p = -10$ results in the best overall performance for change point detection. We also add a small diagonal shift to the normalized Laplacian $\mathbf{L}_{sym}$ to ensure that it is positive definite. For all experiments, we set the shift $\epsilon = log(1 + |p|)$ for $p < 0$ thus no 0 would be encountered in Equation (6.2).

6.5 Method Properties

6.5.1 Evolving Graph Size and Node Permutation

For clarity, we assumed that the number of nodes in multi-view dynamic graphs is constant over time and the same across all views. In many real-world graphs, the number of nodes often changes over time. In practice, MultiLAD can be applied to multi-view dynamic graphs where the graph size changes by considering the top k singular values

from each view before aggregation. In addition, MultiLAD can also adapt to multi-view graphs where the number of nodes in each view differs from each other, again by considering the top k singular values where k is the size of the view with the least number of nodes.

Since the Laplacian spectrum is invariant to row/column permutations, MultiLAD is invariant to node permutations. This implies that the nodes of a dynamic graph are not required to conserve the same ordering for each graph snapshot. This can be very relevant for privacy-sensitive applications where the alignment of nodes across many time steps could cause information leakage. This property can also be very useful in the multi-view setting where aligning node order across views may be technically challenging. In this way, MultiLAD can be applied to a broader range of scenarios than methods relying on a consistent node ordering across snapshots.

6.5.2 Computational Complexity

In a multi-view dynamic graph with T time steps, m views, and an average of $|\mathbf{E}|$ edges per snapshot, MultiLAD has a complexity of $\mathcal{O}(k \cdot m \cdot T \cdot |\mathbf{E}|)$ when computing k singular values. Note that the scalar power mean aggregation is very fast to compute thus computation cost of MultiLAD lies mainly in computing SVD for each view. In real-world networks with hundreds of nodes, MultiLAD is fast to compute, usually taking in the order of minutes on CPUs. For large graphs with millions of nodes where computing SVD is not feasible, efficient approximation techniques such as Network Density of States [45] can be used, as seen in Chapter 5, to approximate the spectrum of the Laplacian.

6.6 Multi-view Experiments

6.6.1 Synthetic Experiments

In this section, we empirically examine the change point detection problem for multi-view dynamic graphs and evaluate our proposed MultiLAD approach. In the synthetic experiments, we focus on answering the following questions: **Q1. Performance Improvement:** How accurately can MultiLAD detect synthetic anomalies by utilizing information from additional views as compared to state-of-the-art single-view baselines and naive multi-view baselines? **Q2. Noise Invariant:** How robust is MultiLAD to noise at individual views and can it leverage multi-view data to reduce the effect of noise? **Q3. Benefit of More Views:** How will the performance of MultiLAD change as we increase the number of incorporated views? and how the performance gain compares to the baselines.

6.6.2 Baselines

Each data point in Figure 6.2, 6.3 and Table 6.3 is the average over 30 trials and the standard deviation is reflected as the shaded area with the same color. All experiments are run on a desktop with AMD Ryzen 5 1600 CPU and 16GB installed RAM. Each MultiLAD trial takes less than 5 minutes. We compare MultiLAD with the following methods:

LAD: We report *the best* average individual view LAD performance over multiple trials using the spectrum of unnormalized and normalized Laplacian ($\mathbf{L}$ and $\mathbf{L}_{sym}$) for comparison (denoted by LAD and NL LAD, respectively). Note that in practice, this is overestimating the performance of such baseline as determining which view is most important is a difficult task.

Naive Aggregation: We consider two naive aggregation strategies to extend LAD into the multi-view setting. First, the maximum of the LAD anomaly score from each view (per time step) is used as the aggregated anomaly score, named maxLAD. Second, the mean of the LAD anomaly scores is used instead and called meanLAD. Lastly, we apply the

same naive aggregation approach to LAD scores obtained from the normalized Laplacian singular values and report them as "NL maxLAD" and "NL meanLAD" respectively.

TENSORSPLAT: A multi-view dynamic graph can be represented as a 4th order tensor $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{T \times M \times N \times N}$ where T, M, N are the total number of time steps, views and nodes respectively. After a rank R CP decomposition of $\boldsymbol{\mathcal{T}}$, one can obtain a signature vector of size R for each time step t. To make this baseline more competitive, we then apply the same sliding window techniques used in LAD to identify the anomalies. Due to its high computational cost, we only report TENSORSPLAT [111] in Section 6.6.4 and average over 10 trials.

6.6.3 Performance Evaluation

Hits at n ($Hits@n$) metric reports the number of correctly identified anomalies out of the top n time steps with the highest anomaly score. For synthetic experiments, we use the ground truth change points or events for evaluation. We extensively examine the performance of MultiLAD as compared to baselines on two widely used graph generative models: the Stochastic Block Model (SBM) [83] and the Barabási-Albert (BA) [6] model. The accuracy reported in plots is the $Hits@7$ score for the 7 planted anomalies. The number of views are set to be 3 except for Section 6.6.6 and 6.6.7. At each time step, the snapshot is re-sampled from its generative model. For all experiments, the sliding windows sizes are $w_s = 5$ and $w_l = 10$ respectively and the startup period $(0, \ldots, w_l)$ is set to have an anomaly score of 0. For experiments involving the SBM model, change points correspond to splitting and merging communities and events are sudden increases in cross-community connectivity.

6.6.4 Case 1: SBM with increasing difficulty

To answer if MultiLAD leverages multi-view information to improve change point detection, we examine a sparse SBM model with an increasing level of difficulty. In a SBM

Table 6.1: Experiment Setting: the changes in the generative model in
(a) Section 6.6.4 and 6.6.6 where the SBM parameter $c_{in} = 12, p_{in} = 0.024$ and $p_{ex} = \frac{c_{out}}{\#nodes}$. The more communities there are, the higher the color intensity is.
(b) Section 6.6.5 where the SBM parameter $p_{in} = 0.024, p_{ex} = 0.004$ and both events and change points are observed.

Change Points: SBM Model parameters		
Time Point	Type	N_c
0	start point	2
16	change point	4
31	change point	6
61	change point	10
76	change point	20
91	change point	10
106	change point	6
136	change point	4

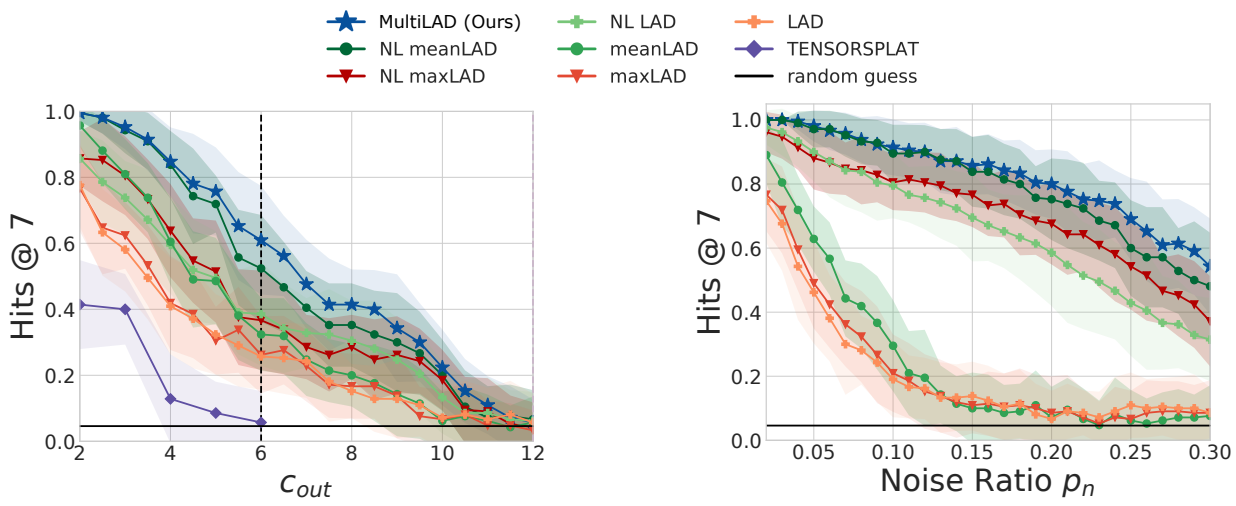
(a) anomalies in Section 6.6.4 and 6.6.6

Events & Change Points: SBM Model parameters				
Time Point	Type	N_c	p_{in}	p_{ex}
0	start point	4	0.024	0.004
16	event	4	0.024	**0.012**
31	change point	**10**	0.024	0.004
61	event	10	0.024	**0.012**
76	change point	**2**	0.024	0.004
91	event	2	0.024	**0.012**
106	change point	**4**	0.024	0.004
136	event	4	0.024	**0.012**

(b) anomalies in Section 6.6.5

model with n nodes, $p_{in} = \frac{c_{in}}{n}$ and $p_{ex} = \frac{c_{out}}{n}$ are the edge probabilities for the within the community and cross-community edges. The constants c_{in} and c_{out} are used to adjust the strength of community connectivity. In this experiment, we set $c_{in} = 12$ and consider different values for c_{out}; note that as $c_{in} - c_{out}$ gets smaller, the task increases in difficulty. As events would be easy to detect when the difference between c_{in} and c_{out} is small, we only consider change points for this experiment (see Table 6.1 (a)).

Figure 6.2 (a) shows that MultiLAD significantly outperforms all baselines as the community structure becomes harder to detect. MultiLAD is able to aggregate over 3 views and significantly outperform the single view method LAD by as much as 48.3% at $c_{out} = 4$. MultiLAD also outperforms the best naive multi-view baseline NL meanLAD by 9.5% at $c_{out} = 7$. Note that for $c_{out} > 11$, the community structure is almost non-existent as the graph closely resembles an Erdős-Rényi graph [51] where all edges are sampled with equal probability. As a reference for the strength of community structure, we indicate the community detection limit for spectral clustering methods on two equal-sized communities as a dotted vertical line in the figure. The limit is reached when $c_{in} - c_{out} = \sqrt{2(c_{in} + c_{out})}$ [143] and in this case, $c_{in} = 12$ and $c_{out} = 6$. We observe that after $c_{out} = 6$,



(a) SBM no noise

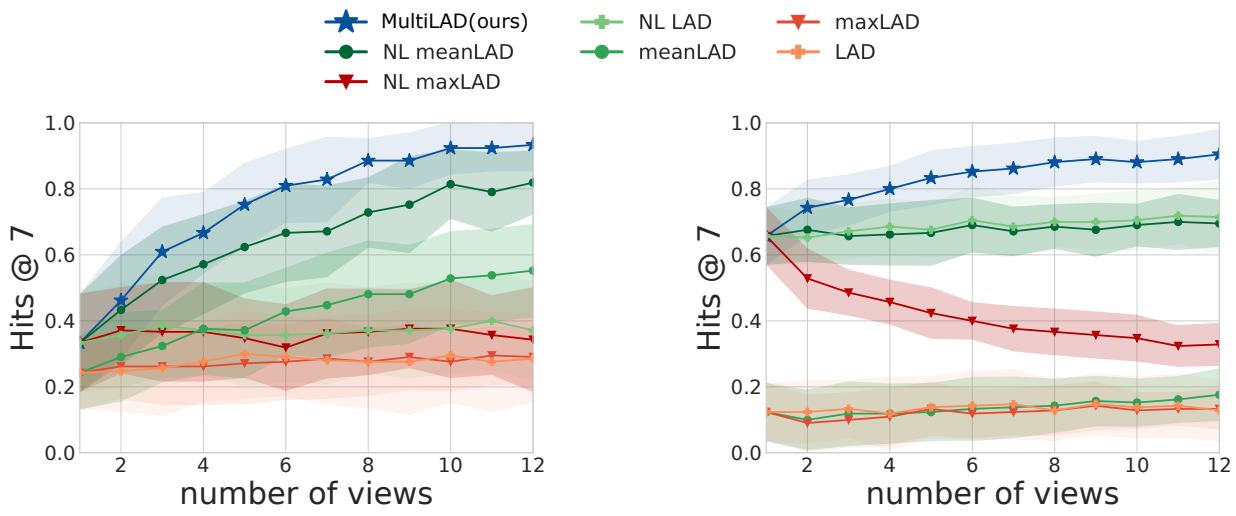
(b) SBM with noise

Figure 6.2: (a) MultiLAD significantly outperforms all baselines as the inter-community connectivity c_{out} approaches the intra-community connectivity $c_{in} = 12$. We also indicate the community detection limit for spectral clustering methods with a dotted vertical line. (b) MultiLAD effectively leverages multi-view information to filter out noise injected into individual views while single-view methods such as LAD is easily disrupted by high amount of noise.

MultiLAD still detects the change points to a significant extent while outperforming other methods. Although being a multi-view baseline, TENSORSPLAT's performance is much worse than MultiLAD and even single-view LAD. Designing an effective multi-view method is an important direction.

6.6.5 Case 2: all views are perturbed by noise

To examine if MultiLAD is robust to noise from individual views, we add noise to the SBM generative process at each view by examining each pair node pair i, j and flip the entry $\mathcal{A}_{i,j}$ of the adjacency matrix to $1 - \mathcal{A}_{i,j}$ with probability p_n. In this experiment, we set $p_{in} = 0.024$ and $p_{ex} = 0.004$ and adjust p_n for the noise level. We also include both events and change points (see Table 4.2 (b)).



(a) SBM **(b)** BA

Figure 6.3: MultiLAD sees the most performance gain from additional views in the SBM model setting (a) and BA model setting (b) when compared to all baselines.

Figure 6.2 (b) demonstrates that MultiLAD is highly robust to noise and outperforms all baselines. Notably, single-view methods like LAD are highly susceptible to noise and their performance degrades quickly as higher ratios of noise are injected, In comparison, MultiLAD maintains strong performance despite each view having a high level of noise. This shows that using the scalar power mean to aggregate singular values from individual views is a very effective way to filter out noise. When $p_n = 0.30$, MultiLAD outperforms single-view LAD by 45.7% and NL LAD by 22.9%. MultiLAD upper bounds all naive multi-view baselines and significantly outperform NL maxLAD and NL meanLAD when the noise ratio is high (i.e. $p_n > 0.15$).

6.6.6 Case 3: SBM with additional views

To understand how MultiLAD benefits from having additional views, we follow the same setting as in Section 6.6.4 and add more views. For this experiment, we fix $p_{in} = 0.024$ and $p_{ex} = 0.012$, and compare the performances of all methods when the number of available views increases.

Change Points: BA Model parameters		
Time Point	Type	m
0	start point	1
16	change point	2
31	change point	3
61	change point	4
76	change point	5
91	change point	6
106	change point	7
136	change point	8

Table 6.2: Experimental setting in Section 6.6.7 where the BA model parameter m is the number of edges to attach from a new node to existing nodes. The increased color intensity in the table indicates a higher m value.

Figure 6.3 (a) shows that MultiLAD benefits the most from having additional views. Single-view methods such as LAD are unable to take advantage of the multi-view data thus remaining at the same performance. Another interesting observation is that max aggregation barely benefits from the additional views as it can be easily dominated by the anomaly score from one particular view. Both NL meanLAD and meanLAD improve with additional views as they compute the average of the anomaly score thus converging to the true expected anomaly score. The strong performance of MultiLAD can be attributed to the fact that MultiLAD directly aggregates the singular values while NL meanLAD and NL maxLAD are dependent on the anomaly score output from individual views. The biggest performance gap appears at 7 views where MultiLAD outperforms NL maxLAD by 46.7% and NL meanLAD by 15.7%.

6.6.7 Case 4: all views are BA models

Lastly, we further investigate MultiLAD performance in the Barabási-Albert (BA) model to show that MultiLAD is effective beyond the SBM model. In this experiment, the change points correspond to the densification of the network (parameter m, increased number of

Table 6.3: MultiLAD outperforms all alternative approaches on different datasets.

Metric	Hits @ 7			
Experiment	SBM (3v)	SBM (noisy)	SBM (12v)	BA (12v)
MultiLAD	**.61 ± .16**	**.86 ± .09**	**.94 ± .08**	**.90 ± .08**
LAD [93]	.26 ± .10	.14 ± .11	.29 ± .13	.13 ± .09
maxLAD	.26 ± .12	.11 ± .09	.29 ± .09	.13 ± .06
meanLAD	.32 ± .11	.10 ± .11	.55 ± .14	.18 ± .08
NL LAD	.39 ± .16	.70 ± .14	.37 ± .15	.71 ± .11
NL maxLAD	.37 ± .15	.77 ± .10	.34 ± .16	.33 ± .07
NL meanLAD	.52 ± .16	.84 ± .11	.82 ± .10	.70 ± .07
TENSORSPLAT [111]	.6 ± .9	N/A	N/A	N/A

edges attached from a new node to an existing node). The details are described in Table 6.2 (c).

Figure 6.3 (b) shows that MultiLAD benefits significantly from more views in the BA setting as well. In particular, with just 2 views, MultiLAD outperforms LAD by 61.9%, NL LAD by 9%, and NL meanLAD by 6.7%. MultiLAD is able to efficiently utilize multi-view information as soon as additional views are available and benefits even more when there are more views. We also observe that NL maxLAD has decreasing performance when the number of views increased. This is likely due to naively aggregating the max anomaly score from all views and picking up the noise from additional views rather than useful information. This shows that naive aggregation strategies can have adverse effects. Also, NL meanLAD performs similarly to single view NL LAD and thus is not able to effectively leverage the multi-view information.

6.6.8 Summary of MultiLAD Results

Table 6.3 summarizes the performance comparison between MultiLAD and other baselines across different experiments. We use k'v' to indicate the number of views in the network. For SBM (3v) and SBM (12v), we report the setting where $p_{in} = 0.024$ and $p_{ex} = 0.012$. For noisy SBM setting, we follow Section 6.6.5 and set noise to 0.15. For BA, we follow the setup in Section 6.6.7.

In all settings, MultiLAD significantly outperforms all baselines. By efficient use of multi-view information, MultiLAD improves upon LAD by as high as 62% on the SBM experiments and 77% on the BA experiment. MultiLAD also sees increased gain as additional views become available, achieving 33% gain when using 12 views instead of 3. This shows that MultiLAD effectively leverages the additional views to achieve stronger performance.

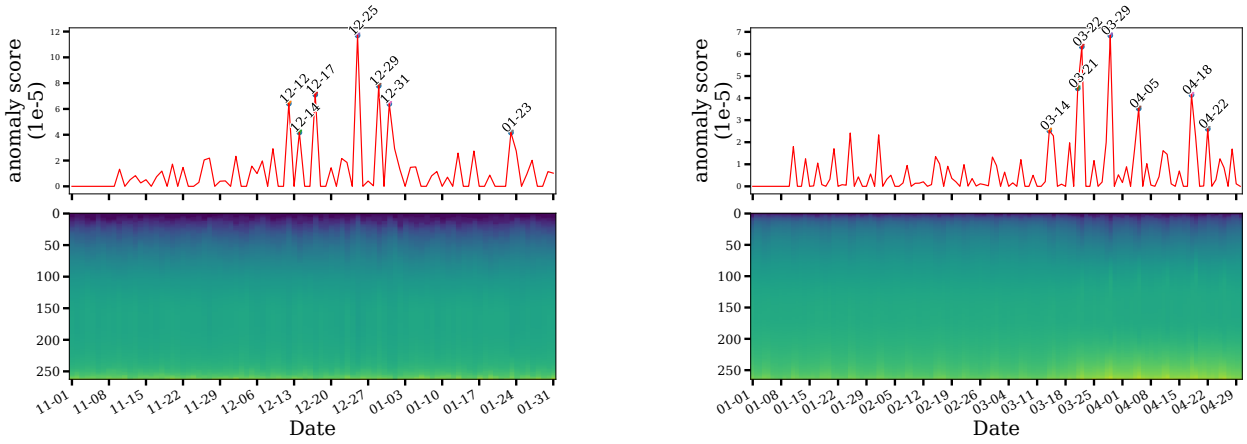
MultiLAD outperforms alternative aggregation strategies including max and mean across all settings and is more robust to noise which shows the benefit of merging information at the signature vector level rather than the anomaly score level. Moreover, max and mean baselines exhibit inconsistent behaviors on different settings. On the BA model, the mean operation does not benefit from having additional views while it is able to outperform single-view baselines on the SBM model. The max operation is detrimental to performance with additional views on the BA model, but not on the SBM model. In contrast, MultiLAD always benefits from additional views.

We can also see that it is better to use the normalized Laplacian (NL LAD) over the unnormalized one (LAD). It is known that on large dense graphs, the distribution of the eigenvalues of the Laplacian is close to the distribution of the degrees of the vertices [199, 79]. After normalization by node degree, the singular values of the normalized Laplacian can better reflect the graph structure, and thus be more suited for change point detection.

6.6.9 Real World Experiments

Next, we examine MultiLAD in the real-world setting. We used New York City (NYC) Taxi & Limousine Commission (NYC TLC)[†] trip records to construct and analyze two dynamic multi-view graphs at different time periods. One of them is a novel multi-view dynamic graph dataset set in 2020, showcasing MultiLAD's ability to detect key dates in the implementation of New York City's COVID-19 response. We use window sizes of 3

[†] https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page



(a) NYC TLC 2015-2016

(b) NYC TLC 2020

Figure 6.4: Anomaly score (top, annotated with the 7 most anomalous time points) and MultiLAD's aggregated spectrum (bottom) for (a). the NYC TLC 2015-2016 dataset and (b).NYC TLC 2020 dataset.

and 7 days in our experiments to model both short-term events and long-term change points.

NYC TLC 2015-2016: From NYC TLC records, we acquired the timestamped trip data from two taxi companies (green and yellow) covering November 2015 to January 2016 similar to [54]. We used these records to construct a dynamic two-view directed weighted graph over 264 nodes (representing taxi zones) in which each time point is a day.

Figure 6.4 shows MultiLAD's anomaly score curve and the corresponding Laplacian spectrum (aggregated using the scalar power mean approach). In the spectrum visualization, the time points with high anomaly scores are visually distinct from their surrounding points thus confirming that MultiLAD's aggregated spectrum correctly retains meaningful information from individual views. Comparing MultiLAD's results to those reported by [54] revealed that there are many high-scoring anomalies in common (including Christmas Day, New Year's Eve, New Year's Day, and the blizzard on January 23rd.) There are some minor differences in the anomalies identified by MultiLAD and SPOTLIGHT which can be attributed to the fact that MultiLAD is analyzing the multi-view version of this dataset while SPOTLIGHT is operating on the single-view version.

NYC TLC 2020: We also extracted NYC TLC data from January 1st 2020 to April 30th 2020. There are 4 views in total with 2 additional views corresponding to for-hire vehicle (FHV; pre-arranged transportation services like community cars and limousines[‡]) and high-volume for-hire vehicle (FHVHV; Uber, Lyft, and Via[§]) transportation services were available for this timecourse.

We hypothesized that anomalies in the 2020 NYC transit dataset would align with key time points in the rollout of NYC's COVID-19 response. Our results suggest this hypothesis was largely correct (see Figure 6.4): MultiLAD is able to identify the adoption of the "NYS on Pause Program" (all non-essential workers must stay home, March 22nd), March 29th (the day after all non-essential construction sites were halted in NYS), and April 18th (the Saturday after the first extension of the "NYS on Pause Program", April 16th). These results show that MultiLAD can indeed detect significant and meaningful real-world events from multiple views. In this case, considering the 4 views individually and determining the most informative one is unpractical.

6.7 Conclusion

In this chapter, we introduced the novel change point detection method: MultiLAD for multi-view dynamic graphs respectively. MultiLAD aggregates the singular values of the normalized Laplacian matrices through the scalar power mean operation and identifies the most informative singular values from each view. On synthetic benchmarks, MultiLAD benefits from additional views, is robust to noise, and outperforms competitive single view and multi-view baselines. On real world traffic networks, we showed that MultiLAD correctly identifies significant events which drastically disrupts the flow of traffic.

[‡] https://www1.nyc.gov/site/tlc/businesses/for-hire-vehicle-bases.page

[§] https://www1.nyc.gov/site/tlc/businesses/high-volume-for-hire-services.page

Chapter 7

Temporal Graph Benchmark for Machine Learning on Temporal Graphs

In the second part of the thesis, we focus on addressing limitations in current evaluation for temporal graph learning methods which lead to over-optimistic performance of methods. These limitations include a lack of diverse and large-scale datasets as well as simplistic evaluation protocols.

Our proposed solution is the *Temporal Graph Benchmark (TGB)*, a collection of challenging and diverse benchmark datasets for realistic, reproducible, and robust evaluation of machine learning models on temporal graphs. TGB datasets are of large scale, spanning years in duration, incorporate both node and edge-level prediction tasks and cover a diverse set of domains including social, trade, transaction, and transportation networks.

For both tasks, we design evaluation protocols based on realistic use cases. We extensively benchmark each dataset and find that the performance of common models can vary drastically across datasets. In addition, on dynamic node property prediction tasks, we show that simple methods often achieve superior performance compared to existing temporal graph models. We believe that these findings open up opportunities for future research on temporal graphs. Finally, TGB provides an automated machine learning pipeline for reproducible and accessible temporal graph research, including data loading,

experiment setup, and performance evaluation. TGB will be maintained and updated on a regular basis and welcomes community feedback. This chapter is based on our publication at the NeurIPS 2023 dataset and benchmark track [94].

Reproducibility: code and datasets in this chapter are available on Github * with documentation website. More information is also available on the TGB website.

7.1 Introduction

Many real-world systems such as social networks, transaction networks, and molecular structures can be effectively modeled as graphs, where nodes correspond to entities and edges are relations between entities. Recently, significant advances have been made for machine learning on static graphs, led by the use of *Graph Neural Networks (GNNs)* [107, 197, 29] and *Graph Transformers* [160, 114, 47], and accelerated by the availability of public datasets and standardized evaluations protocols, such as the widely adopted *Open Graph Benchmark (OGB)* [88].

However, most available graph datasets are designed only for *static* graphs and lack the fine-grained timestamp information often seen in many real-world networks that evolve over time. Examples include social networks [153], transportation networks [34], transaction networks [177] and trade networks [158]. Such networks are formalized as *Temporal Graphs (TGs)* where the nodes, edges, and their features change *dynamically*.

A variety of machine learning approaches tailored for learning on TGs have been proposed in recent years, often demonstrating promising performance [165, 37, 187, 133, 102]. However, Poursafaei *et al.* [158] recently revealed an important issue: these TG methods often portray an over-optimistic performance — meaning they appear to perform better than they would in real-world applications — due to the inherent limitations of commonly used evaluation protocols.

*https://github.com/shenyangHuang/TGB

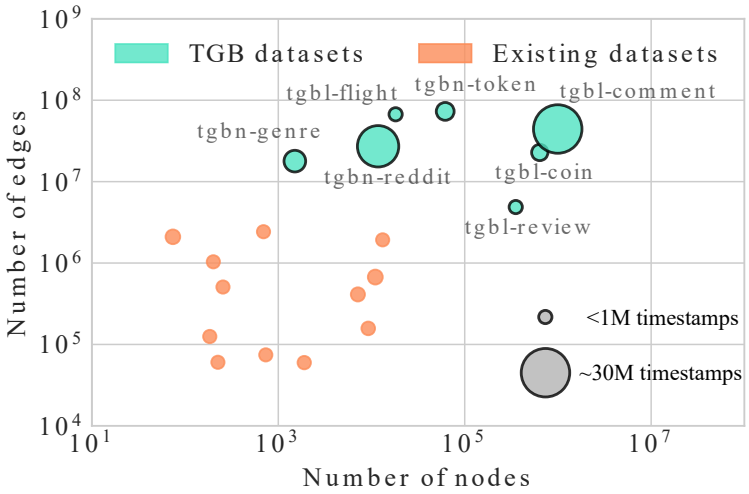


Figure 7.1: TGB consists of a diverse set of datasets that are one order of magnitude larger than existing datasets in terms of number of nodes, edges, and timestamps.

This over-optimism creates serious challenges for researchers. It becomes increasingly difficult to distinguish between the strengths and weaknesses of various methods when their test results suggest similarly high performance. Furthermore, there is a discrepancy between real-world applications of TG methods and the existing evaluation protocols used to assess them. Therefore, there is a pressing need for an open and standardized benchmark that enhances the evaluation process for temporal graph learning, while being aligned with real-world applications.

In this work, we present the *Temporal Graph Benchmark (TGB)*, a collection of challenging and diverse benchmark datasets for realistic, reproducible, and robust evaluation for machine learning on temporal graphs. Figure 7.2 shows TGB's ML pipeline. Inspired by the success of OGB, TGB automates the process of dataset downloading and processing as well as evaluation protocols and allows the user to easily compare their model performance with other models on the public leaderboard. TGB improves the evaluation of temporal graph learning in both *dataset selection* and *evaluation protocol* and covers both edge and node-level tasks.

Dataset Selection. Contrary to real-world networks that typically contain millions of nodes and tens of millions of edges, existing TG benchmark datasets are notably smaller, falling short by several orders of magnitude [158, 187]. Furthermore, these datasets often have limitations in terms of their domain diversity, with a substantial focus on social and interaction networks [165, 115, 133]. This lack of diversity can be problematic as network properties, such as network motifs [140], the scale-free property [21], and the modular structure [146] vary significantly across different domains. Consequently, it is important to benchmark existing methods across a wide variety of domains for a comprehensive evaluation.

To address these limitations, TGB datasets provide diversity in terms of the number of nodes, edges, timestamps, and network domains. As shown in Figure 7.1, TGB datasets are larger in scale and present statistics that were under-explored in prior literature. For instance, the `tgbn-token` dataset has around 73 million edges while the `tgbl-comment` dataset has more than 30 million timestamps. Additionally, TGB introduces four datasets in the novel *node affinity prediction* task to address the scarcity of large-scale datasets for node-level tasks in the current literature.

Improved Evaluation. In TGB, we aim to design the evaluation for both edge and node-level tasks on temporal graphs based on real applications. Historically, the standard approach for dynamic link prediction evaluation is to treat it as a binary classification task using one negative edge per positive edge in the test set [37, 187, 133, 158]. This strategy tends to generate negatives that are easy to predict, given the structure and sparsity of real-world networks [5], leading to inflated model performance estimations [158]. To address this issue, we propose to treat this task as a ranking problem, contrasting each positive sample against multiple negatives and using Mean Reciprocal Rank (MRR) as the metric. Moreover, *historical negatives* – past edges absent in the current step – are generally more difficult to predict correctly than randomly sampled negatives [158]. Thus, we sample both historical and random negatives in link prediction evaluations.

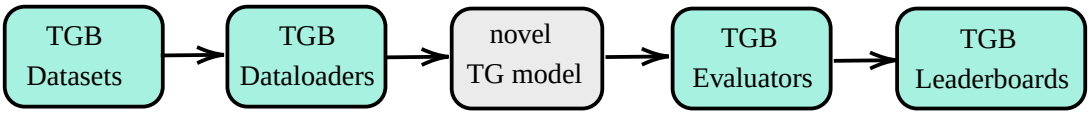


Figure 7.2: Overview of the Temporal Graph Benchmark (TGB) pipeline: (a) TGB includes large-scale and realistic datasets from five different domains with both dynamic link prediction and node property prediction tasks. **(b)** TGB automatically downloads datasets and processes them into `numpy`, `PyTorch` and `PyG` compatible `TemporalData` formats. **(c)** Novel TG models can be easily evaluated on TGB datasets via reproducible and realistic evaluation protocols. **(d)** TGB provides public and online leaderboards to track recent developments in temporal graph learning domain. The code is publicly available as a Python library.

There is a lack of large-scale datasets for node-level tasks in temporal graphs as acquiring dynamic node labels remains challenging due to privacy concerns. We plan to include additional datasets for node classification in the future. As a starting point, we present the novel *node affinity prediction* task, which finds its motivation in recommendation systems. The objective is to anticipate the shifts in user preferences for items over time, as expanded in Section 7.4.2. In this task, node property is considered as the affinity towards different items at a given time. To assess the effectiveness of methods in addressing this task, we adopt the Normalized Discounted Cumulative Gain (NDCG) metric. This metric serves to determine whether the methods' predictions for class importance adhere to the same ordering as the ground truth. In Section 7.6.2, we show that simple heuristics can outperform state-of-the-art TG methods in achieving superior performance for this task.

Public Leaderboard and Reproducible Results. Following the good practice of OGB, TGB also provides an automated and reproducible pipeline for both link and node property prediction tasks. Researchers can submit and compare method performance on the TGB leaderboard.

7.2 Related Work

Temporal Graph Datasets and Libraries. Recently, Poursafaei *et al.* [158] collected six novel datasets for link prediction on continuous-time dynamic graphs while proposing more difficult negative samplings for evaluation. In comparison, we curated seven novel temporal graph datasets spanning both edge and node-level tasks for realistic evaluation of machine learning on temporal graphs. Yu *et al.* [224] presented DyGLib, a platform for reproducible training and evaluation of existing TG models on common benchmark datasets. DyGLib demonstrates the discrepancy of the model performance across different datasets and argues that diverse evaluation protocols of previous works caused an inconsistency in performance reports. Similarly, Skarding *et al.* [185] provided a comprehensive comparative analysis of heuristics, static GNNs, discrete dynamic GNN, and continuous dynamic GNN on dynamic link prediction tasks. They showed that dynamic models outperform their static counterparts consistently and heuristic approaches can achieve strong performance. In all of the above benchmarks, the included datasets only contain a few million edges. In comparison, TGB datasets are orders of magnitude larger in scale in terms of number of nodes, edges, and timestamps. TGB also includes both node and edge-level tasks. Huang *et al.* [96] collected a novel dynamic graph dataset for anomalous node detection in financial networks and compared the performance of different graph anomaly detection methods. In this work, TGB datasets have more edges and timestamps while covering both edge and node tasks.

Temporal Graph Methods. With the growing interest in temporal graph learning, several recent models achieved outstanding performance on existing benchmark datasets. However, due to the limitations of the current evaluation, many methods achieve over-optimistic and similar performance for the dynamic link prediction task [206, 165, 37, 187, 133, 115]. In this work, TGB datasets and evaluation show a clear distinction between SOTA model performance, which helps facilitate future advancement of TG learning methods. Temporal graphs are categorized into discrete-time and continuous-time

temporal graphs [104]. In this work, we focus on the continuous-time temporal graphs as it is more general. Continuous-time TG methods can be divided into node or edge representation learning methods. Node-based models such as *TGN* [165], *DyRep* [195] and *TCL* [205] first leverage the node information such as temporal neighborhood or previous node history to generate node embeddings and then aggregate node embeddings from both source and destination node of an edge to predict its existence. In comparison, edge-based methods such as *CAWN* [206] and *GraphMixer* [37] aim to directly generate embeddings for the edge of interest and then predict its existence. Lastly, the simple memory-based heuristic *EdgeBank* [158] without any learning component has shown surprising performance based on the existing evaluation. We compare these methods on TGB datasets in Section 7.6.

7.3 Temporal Graph Notations

We denote temporal graphs as timestamped edge streams consisting of triplets of source, destination, timestamp; i.e., $\mathcal{G} = \{(s_0, d_0, t_0), (s_1, d_1, t_1), \ldots, (s_T, d_T, T)\}$, where the timestamps are ordered ($0 \leq t_1 \leq t_2 \leq ... \leq T$) [158, 104]. Note that the graph can be directed or undirected, and weighted or unweighted. $\mathcal{G}_t$ is the cumulative graph constructed from all edges in the stream until time t with nodes $\mathbf{V}_t$ and edges $\mathbf{E}_t$. Optionally, there can be node features $\mathbf{X}_t \in \mathbb{R}^{|\mathbf{V}_t| \times k_x}$, where k_x is the size of the node feature vector, and edge features $\mathbf{M}_t \in \mathbb{R}^{|\mathbf{E}_t| \times k_m}$, where k_m is the size of the edge feature vector. We consider a fixed chronological split to form the training, validation, and test set.

7.4 Task Evaluation on Temporal Graphs

Temporal graphs are often used to model networks that evolve over time where nodes are entities and temporal edges are relations between entities through time. In this work, we focus on continuous-time temporal graphs and denote them as timestamped edge

streams consisting of triplets of source, destination, and timestamp; i.e., $\mathcal{G} = \{(s_0, d_0, t_0), (s_1, d_1, t_1), \ldots, (s$ where the timestamps are ordered ($0 \leq t_1 \leq t_2 \leq ... \leq t_T$) [158, 104]. Note that temporal graph edges can have different properties namely being weighted, directed, or attributed. We consider $\mathcal{G}_t$ as the augmented graph of all edges observed in the stream up to the time t with nodes as $\mathbf{V}_t$ and edges as $\mathbf{E}_t$. Optionally, $\mathcal{G}_t$ can contain node features $\mathbf{X}_t \in \mathbb{R}^{|\mathbf{V}_t| \times k_n}$ where k_n is the size of a node feature vector, and edge features $\mathbf{M}_t \in \mathbb{R}^{|\mathbf{E}_t| \times k_m}$ where k_m is the size of an edge feature vector. We consider a fixed chronological split to form the training, validation, and test set.

Evaluation Settings. There are several possible evaluation settings in the temporal graph based on the available information of the test set. In this work, we consider the *streaming setting* where the deployed models need to adapt to new information at inference time. More specifically, we follow the setting in [165] where previously observed test edges can be accessed by the model but back-propagation and weight updates with the test information are not permitted.

7.4.1 Dynamic Link Property Prediction

The goal of *dynamic link property prediction* is to predict the property (oftentimes the existence) of a link between a node pair at a future timestamp. The timeline is chronologically split at two fixed points resulting in three sets of edges E_{train}, $E_{\text{val.}}$, and E_{test}. In TGB, we improve the evaluation setting in the following ways.

Negative edge sampling. In current evaluation [115, 213, 165, 205, 187, 223, 27, 234], only one negative edge is sampled uniformly randomly from all possible node pairs to evaluate against each positive edge. In contrast, in real applications where the true edges are not known in advance, the edges with the highest probabilities predicted by a given model are used to decide which connections should be prioritized. With that in mind, we treat the link prediction task as a ranking problem and sample multiple negative

edges per each positive edge. In particular, for a given positive edge $e^p : (s, d, t)$, we fix the source node s and timestamp t, and sample q different destination nodes. For each dataset, we select the number of negative edges q based on the trade-off between evaluation completeness and the test set inference time.

We sample the negative edges from both the *historical* and *random* negative edges. Historical negatives are sampled from the set of edges that are observed in the training set but are not present at the current timestamp t (*i.e.* $E_t \setminus E_{\text{train}}$), they are shown to be more difficult for models to predict than random negatives [158]. We sample equally from historical and random negative edges. Note that depending on the dataset and the timestamp t, there might not be enough historical negatives to sample from. In this case, we simply increase the ratio of the random negatives to have the desired number of negative edges per positive ones. For reproducibility, we include a fixed set of negatives sampled for each dataset to ensure consistent comparison amongst models.

Performance Metric. The commonly used metric for reporting models' performance for the dynamic link prediction task is either the Area Under the Receiver Operating Characteristic curve (AUROC) or Average Precision (AP). An appropriate metric should be able to capture the ranking of a positive edge amongst the negative ones, which is not fulfilled by either AUROC or AP. Thus, we devise to use of the filtered Mean Reciprocal Rank (MRR) as the evaluation metric for the dynamic link property prediction. The MRR computes the reciprocal rank of the true destination node among the negative or fake destinations.

The MRR varies in the range of $(0, 1]$ and it is a commonly used metric in recommendation systems [211] and knowledge graphs [203, 88, 87]. In addition, recent link prediction literature is also shifting towards adopting the MRR metric [24, 87, 219]. It should be noted that when reporting the MRR, we perform collision checks to ensure that no positive edge is sampled as a negative edge.

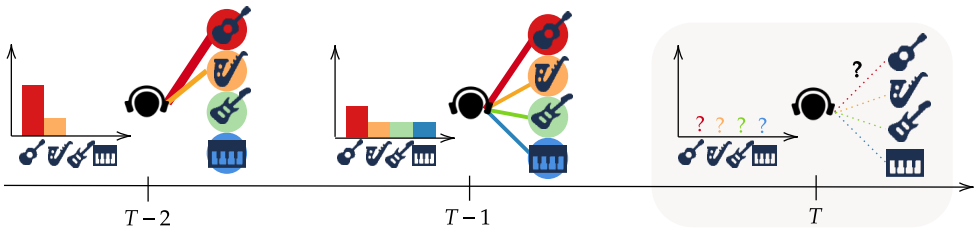


Figure 7.3: The *node affinity prediction* task aims to predict how the preference of a user towards items changes over time. In the `tgbn-genre` example, the task is to predict the frequency at which the user would listen to each genre over the next week given their listening history until today.

7.4.2 Dynamic Node Property Prediction

he goal of *dynamic node property prediction* is to predict the property of a node at any given timestamp t, i.e., to learn a function $f : \mathbf{V}_t \rightarrow \mathcal{Y}$, where $\mathbf{V}_t$ is the set of nodes at time t and $\mathcal{Y}$ is some output space (e.g. $\{-1, +1\}$, $\mathbb{R}$, $\mathbb{R}^p$, etc.). This is a general category for node-level tasks such as node classification and node regression. Here, the property of the node can be a one hot label, a weight vector, or an Euclidean coordinate. Currently, there is a lack of large-scale temporal graph datasets with node labels in the literature; therefore, we first include the *node affinity prediction* task (as defined below). In the future, we will add more node-level tasks and datasets into TGB.

Node affinity prediction. This task considers the affinity of a subset of nodes (representing, e.g., users) towards other nodes (e.g., items) as its property, and how the affinity naturally changes over time. This task is relevant for example in recommendation systems, where it is important to provide personalized recommendations for a user by modeling their preference towards different items over time. Figure 7.3 shows the node affinity prediction task in the context of music recommendation systems as seen in the `tgbn-genre` dataset. In this task, we are given the interaction history of a user with

different music genres, and the goal is to predict the frequency at which the user would listen to each genre over the next week.

More formally, given the observed evolution history of a temporal graph $\mathcal{G}_t$ until current timestamp t, the *node affinity prediction* task (on a dataset such as `tgbn-genre`) predicts the interaction frequency vector $\boldsymbol{y}_t[u,:]$ for a node u over a set of candidate nodes $\mathbf{N}$ within a fixed future period $[t, t+k]$ where k is the window size defined by the application. Each entry in $\boldsymbol{y}_t[u,:]$ corresponds to a candidate node $v \in \mathbf{N}$ and the groundtruth value is generated as follows:

$$\boldsymbol{y}_t[u,v] = \frac{\sum_{t<t_i\leq t+k} w_{(u,v,t_i)}}{\sum_{z\in\mathbf{N}} \sum_{t<t_i\leq t+k} w_{(u,z,t_i)}} \tag{7.1}$$

where $w_{(u,v,t_i)}$ is the weight of the edge (u,v,t_i) (which we assume to be 0 if the edge between u and v is not present at time t_i). Observe that, by definition, $\|\boldsymbol{y}_t[u,:]\|_1 = 1$. We use the Normalized Discounted Cumulative Gain (NDCG) metric that takes into account the relative order of elements. NDCG is commonly used in information retrieval and recommendation systems as a measure of ranking quality [98]. In this work, we use NDCG@10 where the relative order of the top 10 ranked items (*i.e.*, destination nodes) are examined. Specifically in the `tgbn-genre` dataset, the NDCG@10 compares the ground truth to the relative order of the top-10 music genres that a model predicts.

7.5 Datasets

TGB offers nine temporal graph datasets, seven of which are collected and curated for this work. All datasets are split chronologically into the training, validation, and test sets, respectively containing $70\%, 15\%$, and 15% of all edges, in line with similar studies such as [213, 165, 205, 102, 187, 133]. The datasets will be permanently maintained via Digital Research Alliance of Canada funded by the Government of Canada. We consider datasets with more than 5 million edges as medium-size and those with more than 25 million edges as large-size datasets.

Table 7.1: Dataset Statistics. Dataset names are colored based on their scale as small, medium, and large. [P]: Edges can be ***W****eighted*, ***Di****rected*, or ***A****ttributed*.

	Dataset	Domain	# Nodes	# Edges	# Steps	Surprise	Edge Properties[P]
Link	`tgbl-wiki`	interact.	9,227	157,474	152,757	0.108	W: ✘, Di: ✓, A: ✓
	`tgbl-review`	rating	352,637	4,873,540	6,865	0.987	W: ✓, Di: ✓, A: ✘
	`tgbl-coin`	transact.	638,486	22,809,486	1,295,720	0.120	W: ✓, Di: ✓, A: ✘
	`tgbl-comment`	social	994,790	44,314,507	30,998,030	0.823	W: ✓, Di: ✓, A: ✓
	`tgbl-flight`	traffic	18143	67,169,570	1,385	0.024	W: ✘, Di: ✓, A: ✓
Node	`tgbn-trade`	trade	255	468,245	32	0.023	W: ✓, Di: ✓, A: ✘
	`tgbn-genre`	interact.	1,505	17,858,395	133,758	0.005	W: ✓, Di: ✓, A: ✘
	`tgbn-reddit`	social	11,766	27,174,118	21,889,537	0.013	W: ✓, Di: ✓, A: ✘
	`tgbn-token`	transact.	61,756	72,936,998	2,036,524	0.014	W: ✓, Di: ✓, A: ✓

Table 7.1 shows the statistics and properties of the temporal graph datasets provided by TGB. Datasets such as `tgbl-flight`, `tgbl-comment`, `tgbl-coin`, and `tgbn-reddit` are orders of magnitude larger than existing TG benchmark datasets [165, 158, 115], while their number of nodes and edges span a wide spectrum, ranging from thousands to millions. In addition, TGB dataset domains are highly diverse, coming from five distinct domains including social networks, interaction networks, rating networks, traffic networks, and trade networks. Moreover, the duration of the datasets varies from months to years, and the number of timestamps in TGB datasets ranges from 32 to more than 30 million with diverse ranges of time granularity from UNIX timestamps to annually. The datasets can be weighted, directed, or have edge attributes. We also report the *surprise index* (i.e., $\frac{|E_{test} \setminus E_{train}|}{|E_{test}|}$) as defined in [158] which computes the ratio of test edges that are not seen during training. A low surprise index implies that memorization-based methods (such as EdgeBank [158]) can potentially achieve good performance on dynamic link property prediction tasks. We can observe that the surprise index also varies notably across TGB datasets, further contributing to dataset diversity. We discuss the details of TGB datasets next.

`tgbl-wiki`. This dataset stores the co-editing network on Wikipedia pages over one month. The network is a bipartite interaction network where editors and wiki pages are

nodes, while one edge represents a given user who edits a page at a specific timestamp. Each edge has text features from the page edits. The task for this dataset is to predict with which wiki page a user will interact at a given time.

`tgbl-review`. This dataset is an Amazon product review network from 1997 to 2018 where users rate different products in the electronics category from a scale of one to five. Therefore, the network is a bipartite weighted network where both users and products are nodes and each edge represents a particular review from a user to a product at a given time. Only users with a minimum of 10 reviews within the aforementioned time interval are kept in the network. The considered task for this dataset is to predict which product a user will review at a given time.

`tgbl-coin`. This is a cryptocurrency transaction dataset based on the Stablecoin ERC20 transactions dataset [175]. Each node is an address and each edge represents the transfer of funds from one address to another at a time. The network starts on April 1st, 2022, and ends on November 1st, 2022, and contains transaction data of 5 stablecoins and 1 wrapped token. This duration includes the Terra Luna crash where the token lost its fixed price of 1 USD. The considered task for this dataset is to predict with which destination a given address will interact at a given time.

`tgbl-comment`. This dataset is a directed reply network of Reddit where users reply to each other's threads. Each node is a user and each interaction is a reply from one user to another. The network starts from 2005 and ends at 2010. The considered task for this dataset is to predict if a given user will reply to another one at a given time.

`tgbl-flight`. This dataset is a crowd-sourced international flight network from 2019 to 2022. The airports are modeled as nodes, while the edges are flights between airports on a given day. The node features include the type of the airport, the continent where the airport is located, the ISO region code of the airport as well as its longitude and latitude. The edge feature is the associated flight number. In this dataset, our task is to predict whether a flight will happen between two specific airports on a future date. This is useful for foreseeing potential flight disruptions such as cancellations and delays. For instance,

during the COVID-19 pandemic, many flight routes were canceled to combat the spread of COVID-19. In addition, the prediction of the global flight network is also important for studying and forecasting the spread of diseases such as COVID-19 to new regions, as shown in [16, 42].

`tgbn-trade`. This is the international agriculture trading network between nations of the United Nations (UN) from 1986 to 2016. Each node is a nation and an edge represents the sum trade value of all agriculture products from one nation to another one. As the data is reported annually, the time granularity of the dataset is yearly. The considered task for this dataset is to predict the proportion of agriculture trade values from one nation to other nations during the next year.

`tgbn-genre`. This is a bipartite and weighted interaction network between users and the music genres of songs they listen to. Both users and music genres are represented as nodes while an interaction specifies a user listens to a music genre at a given time. The edge weights denote the percentage of which a song belongs to a certain genre. The dataset is constructed by cross-referencing the songs in the *LastFM-song-listens* dataset [115, 81] with that of music genres in the *million-song* dataset [14]. The *LastFM-song-listens* dataset has one month of who-listens-to-which-song information for 1000 users and the *million-song* dataset provides genre weights for all songs in the *LastFM-song-listens* dataset. We only retain genres with at least 10% weights for each song that are repeated at least a thousand times in the dataset. Genre names are cleaned to remove typos. Here, the task is to predict how frequently each user will interact with music genres over the next week. This applies to many music recommendation systems where providing personalized recommendations is important and user preference shifts over time.

`tgbn-reddit`. This is a users and subreddits interaction network. Both users and subreddits are nodes and each edge indicates that a user posted on a subreddit at a given time. The dataset spans from 2008 to 2010. The task considered for this dataset is to learn the interaction frequency towards the subreddits of a user over the next week.

tgbn-token. This is a user and cryptocurrency token transaction network. Both users and tokens are nodes and each edge indicates the transaction from a user to a token. The edge weights indicate the amount of token transferred and considering the disparity between weights, we normalized the edge weights using logarithm. The goal here is to predict how frequently a user will interact with various types of tokens over the next week. The dataset is extracted and curated from this source [175].

7.6 Experiments

For dynamic link property prediction, we include DyRep [195], TGN [165], CAWN [206], TCL [205], GraphMixer [37], NAT [133], TGAT [213] and two deterministic heuristics namely EdgeBank$_{\text{tw}}$ and EdgeBank$_{\infty}$ [158]. For dynamic node property prediction, we include DyRep, TGN, and deterministic heuristics such as persistence forecast [169] and moving average [149]. For the experimental results, we report the average and standard deviation across 5 different runs. We highlight the best results in bold and underline the second-place results.

7.6.1 Dynamic Link Property Prediction

Table 7.2a shows the performance of TG methods for dynamic link property prediction on the `tgbl-wiki` dataset. `tgbl-wiki` is an existing dataset where many methods achieve over-optimistic performance in the literature [165, 206, 37]. With TGB's evaluation protocol, there is now a clear distinction in model performance and NAT achieves the best result on this dataset. As `tgbl-wiki` is the smallest dataset in this task, it is computationally feasible to sample all possible destinations of a given source node. Thus, we compare the true destination with all possible negative destinations in this dataset. In Table 7.2b, we report the results on `tgbl-review` where we sample 100 negative edges per positive edge. Here, we observe that many of the best-performing methods on `tgbl-wiki` have a significant drop in performance including NAT, CAWN and Edgebank. More notably,

Table 7.2: Results for *dynamic link property prediction* on small datasets.

Method	MRR Validation	MRR Test
DyRep [195]	0.072±0.009	0.050±0.017
TGN [165]	0.435±0.069	0.396±0.060
CAWN [206]	0.743±0.004	0.711± 0.006
TCL [205]	0.198±0.016	0.207±0.025
GraphMixer [37]	0.113±0.003	0.118±0.002
TGAT [213]	0.131±0.008	0.141±0.007
NAT [133]	**0.773**±0.011	**0.749**±0.010
EdgeBank$_{tw}$ [158]	0.600	0.571
EdgeBank$_{\infty}$ [158]	0.527	0.495

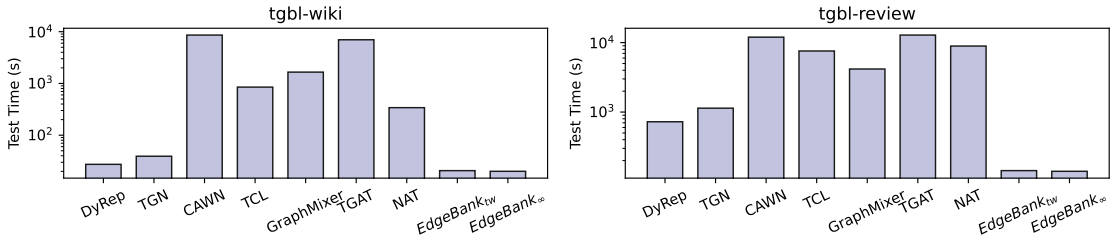
(a) `tgbl-wiki` dataset with *surprise index*: 0.108 (with *all* negative edges per each positive).

Method	MRR Validation	MRR Test
DyRep [195]	0.216±0.031	0.220±0.030
TGN [165]	0.313±0.012	0.349±0.020
CAWN[206]	0.200±0.001	0.193±0.001
TCL [205]	0.199±0.007	0.193±0.009
GraphMixer [37]	**0.428**±0.019	**0.521**±0.015
TGAT [213]	0.324±0.006	0.355±0.012
NAT [133]	0.302±0.011	0.341±0.020
EdgeBank$_{tw}$ [158]	0.0242	0.0253
EdgeBank$_{\infty}$ [158]	0.0229	0.0229

(b) `tgbl-review` dataset with *surprise index*: 0.987 (with *100* negative edges per each positive).

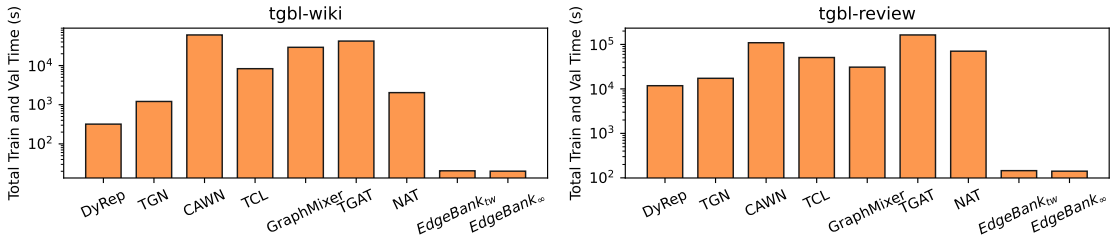
the method rankings also changed significantly with GraphMixer and TGAT being the top two methods. This observation emphasizes the importance of dataset diversity when benchmarking TG methods.

One explanation for the significant performance reduction of some methods on `tgbl-review` is that it has a higher *surprise index* compared to `tgbl-wiki` (see Table 7.1). The surprise index reflects the ratio of edges in the test set that have not been seen during training. Therefore, a dataset with a high surprise index requires more inductive reasoning, as most of the test edges are unobserved during training. As a heuristic that memorizes past edges, EdgeBank performance is inversely correlated with the surprise index and it achieves higher performance when the surprise index of the dataset is low. An interesting future direction is the investigation of the performance of certain categories of methods with the surprise index. For example, the top two methods NAT and CAWN on `tgbl-wiki` both utilizes features from the joint neighborhood of nodes in the queried edge [133]. On the `tgbl-review` dataset, the best performing TGAT is designed for inductive representation learning on temporal graph [213] which fits the inductive nature of `tgbl-review` which has a high surprise index.



(a) `tgbl-wiki` dataset. **(b)** `tgbl-review` dataset.

Figure 7.4: Test set inference time of TG methods can have up to two orders of magnitude difference.



(a) `tgbl-wiki` dataset. **(b)** `tgbl-review` dataset.

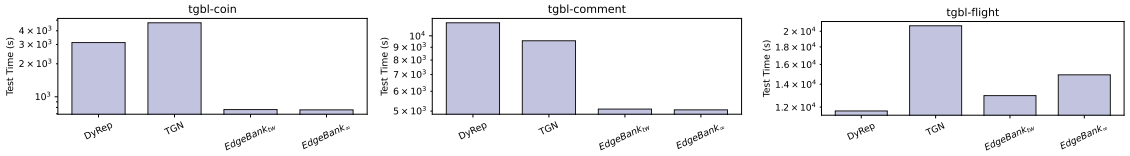
Figure 7.5: Total train and validation time of TG methods can have two orders of magnitude difference.

Figure 7.4a and 7.4b show the inference time of different methods for the test set of `tgbl-wiki` and `tgbl-review`, respectively. Similarly, Figure 7.5a and 7.5b show the total training and validation time of TG methods for `tgbl-wiki` and `tgbl-review`. Notice that as a heuristic baseline, EdgeBank inference, train, or validation time is generally at least one order of magnitude lower than neural network-based methods. We also observe an order of difference in inference time within TG methods. We believe one important future direction is to improve the computational time of these models to be closer to baselines such as EdgeBank, which can better scale to large real-world temporal graphs.

Table 7.3 shows the performance of TG methods on medium and large TGB datasets. Note that some methods, including CAWN, TCL, and GraphMixer, ran out of memory on GPU for these datasets, thus their performance is not reported. Overall, TGN has the

Method	tgbl-coin MRR Validation	tgbl-coin MRR Test	tgbl-comment MRR Validation	tgbl-comment MRR Test	tgbl-flight MRR Validation	tgbl-flight MRR Test
DyRep [195]	0.512±0.014	0.452±0.046	0.291±0.028	0.289±0.033	0.573±0.013	0.556±0.014
TGN [165]	**0.607**±0.014	**0.586**±0.037	**0.356**±0.019	**0.379**±0.021	**0.731**±0.010	**0.705**±0.020
EdgeBank$_{tw}$ [158]	0.492	0.580	0.124	0.149	0.363	0.387
EdgeBank$_{\infty}$ [158]	0.315	0.359	0.109	0.129	0.166	0.167

Table 7.3: Results for *dynamic link property prediction* task on medium and large datasets.



(a) tgbl-coin dataset. **(b)** tgbl-comment dataset. **(c)** tgbl-flight dataset.

Figure 7.6: Inference time comparison of TG methods on medium and large datasets.

best performance on all of these three datasets. Surprisingly, the EdgeBank heuristic is highly competitive on the tgbl-coin dataset where it even significantly outperforms DyRep. Therefore, it is important to include EdgeBank as a baseline for all datasets. Another observation is that for medium and large TGB datasets, there can be a significant performance change for a single model between the validation and test set. This is because TGB datasets span over a long time (such as tgbl-comment, lasting 5 years) and one can expect that models need to deal with potential distribution shifts between the validation set and the test set. Figure 7.6a, 7.6b and 7.6c reports the test time for TG methods on tgbl-coin, tgbl-flight and tgbl-comment, respectively. On both tgbl-coin and tgbl-comment, Edgebank is at least one order of magnitude faster than TGN and DyRep while on the tgbl-flight, due to the large number of temporal edges, DyRep is the fastest method.

Method	tgbn-trade NDCG@10 Validation	tgbn-trade NDCG@10 Test	tgbn-genre NDCG@10 Validation	tgbn-genre NDCG@10 Test	tgbn-reddit NDCG@10 Validation	tgbn-reddit NDCG@10 Test	tgbn-token NDCG@10 Validation	tgbn-token NDCG@10 Test
DyRep [195]	0.394±0.001	0.374±0.001	0.357±0.001	0.351±0.001	0.344± 0.001	0.312± 0.001	0.151± 0.006	0.141± 0.006
TGN [165]	0.395±0.002	0.374±0.001	0.403±0.010	0.367±0.058	0.379± 0.004	0.315± 0.020	0.189± 0.005	0.169± 0.006
persistence Fore. [169]	**0.860**	**0.855**	0.350	0.357	0.380	0.369	0.403	0.430
Moving Avg. [149]	0.841	0.823	**0.499**	**0.509**	**0.574**	**0.559**	**0.491**	**0.508**

Table 7.4: *Node affinity prediction* results.

7.6.2 Dynamic Node Property Prediction

Table 7.4 shows the performance of various methods on the node affinity prediction task in the dynamic node property prediction category. As node-level tasks have received less attention compared to edge-level tasks in the literature, adopting methods that are specially designed for link prediction to this task is non-trivial. As a result, these methods are omitted in this section. Considering Table 7.4, the key observation is that simple heuristics like persistence forecast and moving average are strong contenders to TG methods such as DyRep and TGN. Notably, persistence forecast is SOTA on `tgbn-trade` while moving average is the best performing on other datasets. TGN is second place on `tgbn-genre` dataset. Different from link prediction where the existence of a link is cast as binary classification, the node affinity prediction task compares the likelihood or weight that the model assigns to different target nodes (mostly positive links). These results highlight the need for the future development of TG methods that can acquire flexible node representations capable of learning how user preferences evolve over time.

7.7 Conclusion

To enable realistic, reproducible, and robust evaluation for machine learning on temporal graphs, we present the Temporal Graph Benchmark in this chapter, a collection of challenging and diverse datasets. TGB datasets are diverse in their dataset properties as well as being orders of magnitude larger than existing ones. TGB includes both *dynamic link property prediction* and *dynamic node property prediction* tasks, while providing an au-

tomated pipeline for researchers to evaluate novel methods and compare them on the TGB leaderboards. In dynamic link property prediction, we find that model rankings can vary significantly across datasets, thus demonstrating the necessity to evaluate on the diverse range of TGB datasets. Surprisingly for dynamic node property prediction, simple heuristics such as persistence forecast and moving average outperforms SOTA methods such as TGN. This motivates the development of more TG methods for node-level tasks.

Impact on Temporal Graph Learning. Significant advancements in machine learning are often accelerated by the availability of public and well-curated datasets such as ImageNet [40] and OGB [88]. Since the release of TGB, TGB has become a common and standard benchmark for temporal graph learning, facilitating novel methodological changes and benchmarking state-of-the-art methods. We plan to update TGB regularly with community feedback as well as adding additional datasets and tasks.

Chapter 8

TGB 2.0: A Benchmark for Learning on Temporal Knowledge Graphs and Heterogeneous Graphs

In the previous chapter, we introduced the temporal graph benchmark, a robust and challenging benchmark for homogeneous temporal graphs. However, many real world data are naturally formulated as multi-relational temporal graphs, i.e. graphs that contains edges describing distinct relations between nodes. Recently, many novel models have been proposed for ML on such graphs intensifying the need for robust evaluation and standardized benchmark datasets. However, the availability of such resources remains scarce and evaluation faces added complexity due to reproducibility issues in experimental protocols.

To address these challenges, we introduce Temporal Graph Benchmark 2.0 (TGB 2.0), a novel benchmarking framework, extending the Temporal Graph Benchmark, tailored for evaluating methods for predicting future links on Temporal Knowledge Graphs and Temporal Heterogeneous Graphs with a focus on large-scale datasets. TGB 2.0 facilitates comprehensive evaluations by presenting eight novel datasets spanning five domains with up to 53 million edges. TGB 2.0 datasets are significantly larger than existing datasets in

terms of number of nodes, edges, or timestamps. In addition, TGB 2.0 provides a reproducible and realistic evaluation pipeline for multi-relational temporal graphs. Through extensive experimentation, we observe that 1) leveraging edge-type information is crucial to obtain high performance, 2) simple heuristic baselines are often competitive with more complex methods, and 3) most methods fail to run on our largest datasets, highlighting the need for research on more scalable methods. This chapter is based on our work recently accepted to the NeurIPS 2024 Datasets and Benchmarks Track [63].

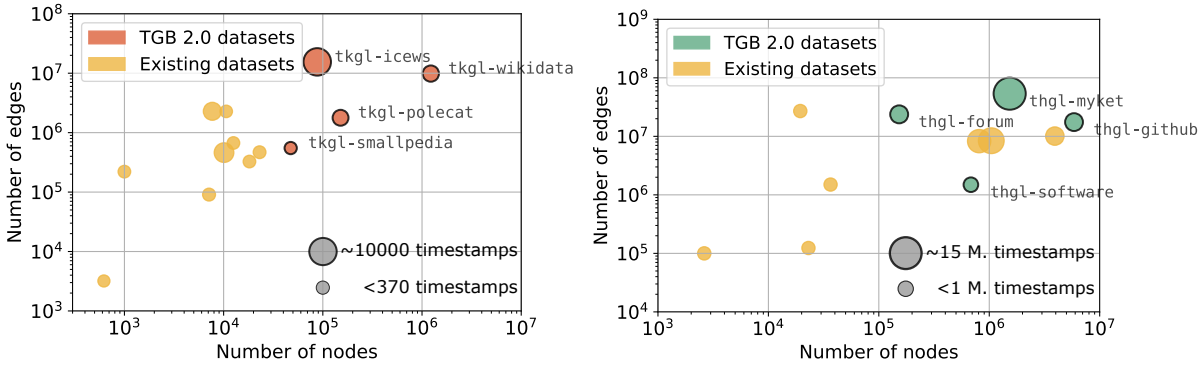
Reproducibility. TGB 2.0 code and datasets are available on Github * and the TGB 2.0 website provides detailed documentation.

8.1 Introduction

Learning from graph-structured data has become ubiquitous in many applications such as recommendation systems [115, 211], knowledge base completion [144, 127] and molecular learning [160, 7]. Relational data often evolves over time and can contain multiple types of relations. These complex interactions and temporal dependencies can be captured by *multi-relational* temporal graphs. In recent years, various approaches have emerged to predict future links in such graphs, notably for prediction on Temporal Knowledge Graphs (TKGs) [124, 128] and Temporal Heterogeneous Graphs (THGs) [120, 100]. These approaches capture the rich information from multi-relational data, developing distinct lines of research from that of single-relational temporal graphs [165, 133]. However, benchmarking on multi-relational temporal graphs faces two main challenges: *inconsistent evaluation* and *limited dataset size*.

Inconsistent Evaluation. Evaluation for multi-relational temporal graphs faces significant challenges. Recently, it was shown that existing evaluation for TKGs has inconsistencies in a) the evaluation metrics, b) mixing multi-step and single-step prediction

*https://github.com/shenyangHuang/TGB.



(a) Novel Temporal Knowledge Graphs **(b)** Novel Temporal Heterogeneous Graphs

Figure 8.1: Existing benchmark datasets (yellow) vs. novel datasets in TGB 2.0 for TKG (a) marked in orange and THG (b) marked in green. Circle sizes correspond to the number of timestamps. TGB 2.0 datasets are significantly larger than existing datasets in number of nodes, edges and timestamps.

settings and c) using different versions of the same dataset [65]. Similar inconsistencies have been observed in related areas such as link prediction on static knowledge graphs [190, 164], node and graph classification on static graphs [178, 52], and temporal graph link prediction [94]. In addition, for link prediction on THGs, existing evaluation often includes a single random negative per positive edge [120, 216], leading to over-optimistic performances, inconsistent evaluation, and reducing performance differentiation between methods [158].

Limited Dataset Size. Existing evaluations are conducted on predominantly small-scale datasets. For example, commonly used TKG and THG datasets consists of less than two million edges and one million nodes [120, 100, 124, 122]. However, real-world networks typically contain tens of millions of nodes and edges thus existing datasets rarely reflect the true scale of datasets in practice. In addition, significant efforts were made to design scalable graph learning methods for real applications which require the availability of large-scale datasets [88, 87, 94]. These challenges hinder meaningful comparisons between methods and the accurate assessment of progress, hamper advancements in the field. Therefore, there is an urgent need for a public and standardized benchmark to

facilitate proper and fair comparison between methods, accelerating research for multi-relational temporal graphs.

To address the aforementioned challenges, we present TGB 2.0, a novel benchmark designed for future link prediction on multi-relational temporal graphs. Building upon the foundations of the Temporal Graph Benchmark (TGB) [94] where only *single-relation temporal graphs* are included, TGB 2.0 introduces *multi-relational temporal graph* datasets. TGB 2.0 adds four novel TKG datasets and four novel THG datasets of varying scale, spanning five domains. Figure 8.1 shows the difference in scale of the novel datasets in TGB 2.0 when compared to existing ones. Figure 8.1a shows that TGB 2.0 TKG datasets (marked in orange) are orders of magnitude larger than existing ones in terms of the number of nodes, edges, and timestamps. Figure 8.1b shows that TGB 2.0 THG datasets (marked in green) are significantly larger than existing datasets. With `thgl-myket` dataset quintupling the number of edges and timestamps while `thgl-github` has the most number of nodes to date. Additionally, TGB 2.0 provides an automated evaluation pipeline for reproducible and realistic evaluation on multi-relational temporal graphs. In TGB 2.0, the dynamic link property prediction task is treated as a ranking problem where multiple negative edges are ranked against the positive edge. For large datasets, we sample negative edges based on the edge type of the query, thus closely approximates the complete evaluation where all negative edges are used. Overall, TGB 2.0 presents a benchmark for realistic, challenging, and reproducible evaluation on *multi-relational temporal graphs* while providing an automated pipeline for dataset downloading, processing, and evaluation as well as a public leaderboard.

TGB 2.0 has the following main contributions:

- **Large and diverse datasets for multi-relational graphs.** We present four novel TKGs that are orders of magnitude larger than existing ones and four novel THGs that are significantly larger in number of nodes, edges and timestamps when compared with current ones.

- **Realistic and reproducible evaluation.** We provide an evaluation pipeline for multi-relational temporal graphs, which automates the dataset downloading, processing, and benchmarking process for seamless reproducibility. TGB 2.0 evaluation uses the ranking metric MRR and samples challenging negative edges based on the edge type information, hence providing realistic evaluation.

- **Experimental insights.** The main insight from our experiments is that for both THGs and TKGs, all methods (apart from heuristics) fail to scale to our largest datasets, highlighting the need for more research on scalable methods. Surprisingly, the heuristic baselines perform competitively with more sophisticated methods. On THG datasets, we observe that methods that leverage the edge type and node type information achieve strong performance. Finally, across TKG datasets, we observe a strong correlation between the recurrency degree of a given relation type and the performance of methods, suggesting large room for improvement on low recurrency relations.

8.2 Related Work

TKG Methods. Most TKG forecasting methods utilize a discrete-time representation, except for [76]. Some methods integrates the message-passing paradigm from static graphs [172, 139] with sequential techniques [103, 124, 74, 75, 122, 126]. Other approaches combine Reinforcement Learning with temporal reasoning for future link prediction [123, 189]. Rule-based methods [128, 108, 137, 129, 125] employ strategies to learn temporal logic rules while others [235, 215, 228] combines a blend of different methodologies. For TKG forecasting, common benchmark datasets include YAGO [135], WIKI [116, 103], GDELT [117] and the Integrated Crisis Early Warning System (ICEWS) dataset [18]. However, these datasets are orders of magnitude smaller than our TKG datasets in number of nodes, edges and timestamps. In `tkgl-icews`, we include the full ICEWS dataset [18] spanning 28 years when comparing to prior versions containing only one or a few years [62, 103, 43].

Similarly, our `tkgl-wikidata` dataset is orders of magnitude larger than the existing WIKI dataset [64, 124] in size of nodes, edges and timestamps.

THG Methods. THG methods can be categorized based on their time representation: discrete-time methods and continuous-time methods. Examples of continuous time methods include HTGN-BTW [226] and STHN [120]. HTGN-BTW [226], enabling TGN [165] to accommodate heterogeneous node and edge types. STHN [120] utilizes a link encoder and patching techniques to incorporate edge type and time information respectively. Discrete-time methods includes random walk based methods [15, 220] and message-passing based methods[216, 41]. However, it is difficult to adapt discrete-time methods for continuous-time datasets. Common THG datasets such as MathOverflow [152], Netflix [11] and Movielens [77] are small and only contain a few million edges [120]. Large datasets such as Dataset A and B from the WSDM 2022 Challenge and the TRACE and THEIA datasets [163] are only evaluated with one negative sample per positive edge, which is shown to lead to over-optimistic and insufficient evaluation [94, 158]. Here, we introduce the large `thgl-myket` dataset with 53 million edges and 14 million timestamps.

Graph Learning Benchmarks. The Open Graph Benchmark (OGB) [88] and the OGB large scale challenge [87] are popular benchmarks accelerating progress on static graphs. Recently, the Temporal Graph Benchmark was introduced for temporal graph learning, consisting of large datasets for single-relation temporal graphs [94]. In this work, we introduce novel TKG and THG datasets, incorporating multi-relational temporal graphs into TGB. Recently, detailed performance comparisons for deep learning methods on dynamic graphs are conducted in [68], however, multi-relational temporal graph datasets were not included in the comparison. While efforts like [65] have highlighted evaluation inconsistencies in TKG, their study focuses on existing smaller-scale datasets where no novel evaluation framework was proposed. Moreover, recent findings by [64] reveal that a simple heuristic baseline outperforms existing methods on some datasets, thus un-

derscores the necessity for comparison with baselines. In this work, TGB 2.0 includes four novel TKG and four novel THG datasets as well as a standardized and reproducible evaluation pipeline.

8.3 Datasets

TGB 2.0 introduces eight novel datasets from five distinct domains consisting of four TKGs and four THGs. We split all datasets chronologically into training, validation, and test sets, respectively containing 70%, 15%, and 15% of all edges in line with existing studies [94, 165, 133] and ensure that edges of a timestamp can only exist in either train or validation or test set. The datasets will be permanently maintained via the Digital Research Alliance of Canada (funded by the Government of Canada).

Dataset Details. Here we describe each TGB 2.0 dataset in detail. Temporal Knowledge Graph datasets start with the prefix `tkgl-` while Temporal Heterogeneous Graph datasets start with `thgl-`.

`tkgl-smallpedia`. This TKG dataset is constructed from the Wikidata Knowledge Graph [201] where it contains facts paired with Wikipedia pages. Each fact connects two entities via an explicit relation (edge type). This dataset contains Wikidata entities with IDs smaller than 1 million. The temporal relations either describe point-in-time relations (event-based) or relations with duration (fact-based). We also provide static relations from the same set of Wikidata pages which include 978,315 edges that can be used to enhance model performance. The task is to predict future facts.

`tkgl-polecat`. This TKG dataset is based on the POLitical Event Classification, Attributes, and Types (POLECAT) dataset [171] which records coded interactions between socio-political actors of both cooperative or hostile actions. POLECAT utilizes the PLOVER ontology [73] to analyze new stories in seven languages across the globe to generate time-stamped, geolocated events. These events are processed automatically via NLP

tools and transformer-based neural networks. This dataset records events from January 2018 to December 2022. The task is to predict future political events between political actors.

tkgl-icews. This TKG dataset is extracted from the ICEWS Coded Event Data [18, 180] which spans a time frame from 1995 to 2022. The dataset records political events between actors. It is classified based on the CAMEO taxonomy of events [66] which is optimized for the study of mediation and contains a number of tertiary sub-categories specific to mediation. When compared to PLOVER ontology in tkgl-polecat, the CAMEO codes have more event types (391 compared to 16). The task is to predict future interactions between political actors.

tkgl-wikidata. This TKG dataset is extracted from the Wikidata KG [201] and constitutes a superset of tkgl-smallpedia. The temporal relations are properties between Wikidata entities. tkgl-wikidata is extracted from wikidata pages with IDs in the first 32 million. We also provide static relations from the same set of Wiki pages containing 71,900,685 edges. The task is to forecast future properties between wiki entities.

thgl-software. This THG dataset is based on Github data collected by GH Arxiv. Only nodes with at least 10 edges were kept in the graph, thus resulting in 14 types of relations and 4 node types (similar relations to [2]). The dataset spans January 2024. The task is to predict the next activity of a given entity, e.g., which pull request the user will close at a given time.

thgl-forum. This THG dataset is based on the user and subreddit interaction network on Reddit [141]. The node types encode users or subreddits, the edge relations are "user reply to user" and "user -post" in subreddits. The dataset contains interactions from January 2014. The task is to predict which user or subreddit a user will interact with at a given time.

thgl-myket. This THG dataset is based on the Myket Android App market. Each edge documents the user installation or updates interaction within the Myket market. The data spans six months and two weeks and when compared to an existing smaller

version [130], this dataset contains the full data without downsampling. Overall, the dataset includes information on 206,939 applications and over 1.3 million anonymized users from June 2020 to January 2021.

`thgl-github`. This THG dataset is based on Github data collected from the GH Arxiv. This is a large dataset from a different period from `thgl-software`. We extract user, pull request, issue and repository nodes and track 14 edge types. The nodes with two or fewer edges are filtered out. The dataset contains the network as of March 2024. The task is to predict the next activity of an entity.

Varying Scale. Table 8.1 shows the detailed characteristics of all datasets, such as the number of quadruples and nodes. TGB 2.0 datasets vary significantly in scale for the number of nodes, edges, and time steps. We observe an increase in runtime and memory requirements from `tkgl-smallpedia` to `tkgl-polecat` to `tkgl-icews` and `tkgl-wikidata`. In practice, these requirements depend on the combination of the number of nodes, edges, and time steps. To account for such benchmarking requirements, we categorize the datasets into small, medium, and large datasets. Small datasets are suitable for prototyping methods, while medium and large datasets test method performance at increasingly large scales.

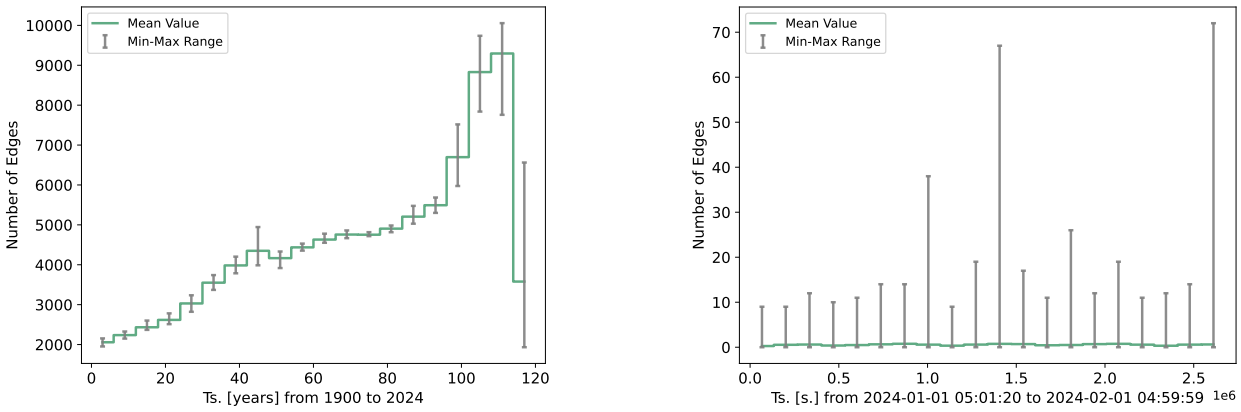
Table 8.1 reports dataset statistics: the *Proportion of Inductive Test Nodes (Induct. Test Nodes)* is the proportion of nodes in the test set that have not been seen during training. The *Recurrency Degree (Rec)*, which is defined as the fraction of test temporal triples (s, r, o, t^+) for which there exists a $k < t^+$ such that $(s, r, o, k) \in G$. The *Direct Recurrency Degree (DRec)* which is the fraction of temporal triples (s, r, o, t^+) for which it holds that $(s, r, o, t^+ - 1) \in G$ [64]. Also, we represent a novel metric called *Consecutiveness Value (Con)*, which quantifies if a given temporal triple repeats at consecutive timestamps by averaging the maximum number of consecutive timesteps during which a triple holds true across all triples in the dataset. Intuitively, fact-based relations which are true across multiple consecutive time steps will result in a higher *Consecutiveness Value*.

Table 8.1: Dataset information including common statistics and the proportion of Inductive Test nodes (Induct. Test Nodes), the Direct Recurrency Degree (DRec), the Recurrency Degree (Rec), the Consecutiveness Value (Con), as well as the mean number of edges and nodes per timestep (Mean Edges/Ts. and Mean Nodes/Ts.)

	Temporal Knowledge Graphs (tkgl-)				Temporal Heterogeneous Graphs (thgl-)			
Dataset	smallpedia	polecat	icews	wikidata	software	forum	github	myket
Domain	knowledge	political	political	knowledge	software	social.	software	interac.
# Quadruples	550,376	1,779,610	15,513,446	9,856,203	1,489,806	23,757,707	17,499,577	53,632,788
# Nodes	47,433	150,931	87,856	1,226,440	681,927	152,816	5,856,765	1,530,835
# Edge Types	283	16	391	596	14	2	14	2
# Node Types	-	-	-	-	4	2	4	2
# Timesteps	125	1,826	10,224	2,025	689,549	2,558,457	2,510,415	14,828,090
Granularity	year	day	day	year	second	second	second	second
Induct. Test Nodes	0.26	0.12	0.05	0.34	0.13	0.02	0.14	0.01
DRec	0.71	0.07	0.11	0.61	0.00	0.00	0.00	0.00
Rec	0.72	0.43	0.63	0.61	0.10	0.63	0.01	0.37
Con	5.82	1.07	1.14	5.05	1.00	1.00	1.00	1.00
Mean Edges/Ts.	4,403.01	974.59	1,516.91	4,867.26	0.56	8.87	6.54	3.15
Mean Nodes/Ts.	5,289.16	550.60	1,008.65	5,772.16	0.86	12.96	9.77	6.24

Diverse Statistics. TGB 2.0 datasets exhibits diverse dataset statistics. For example, `tkgl-wikidata`, `tkgl-smallpedia`, `tkgl-polecat`, and `thgl-software` all have more than 10% test nodes that are inductive (i.e. nodes unseen in the training set), thus testing the inductive capability of methods. Variations in the recurrence of relations are evident with `tkgl-smallpedia` and `tkgl-wikidata` showing higher Recurrency Degrees compared to others. The DRec highlights the disparities between THG and TKG datasets, where the finer, second-wise time granularity of THG leads to a DRec of 0 implying no repetition of facts across subsequent time steps. On TKG datasets, the high Consecutiveness Value for `tkgl-smallpedia` and `tkgl-wikidata` exhibit a prevalence of long-lasting facts, contrasting with `tkgl-icews` and `tkgl-polecat` which documents political events. In comparison, THG datasets describe one-time events, thus displaying lower Con values.

Figure 8.2 shows the number of edges per timestamp for `tkgl-smallpedia` and `thgl-software`, reported over twenty bins with bars showing the min/max in each bin. Figure 8.2 underscores distinctions between datasets, particularly in terms of time

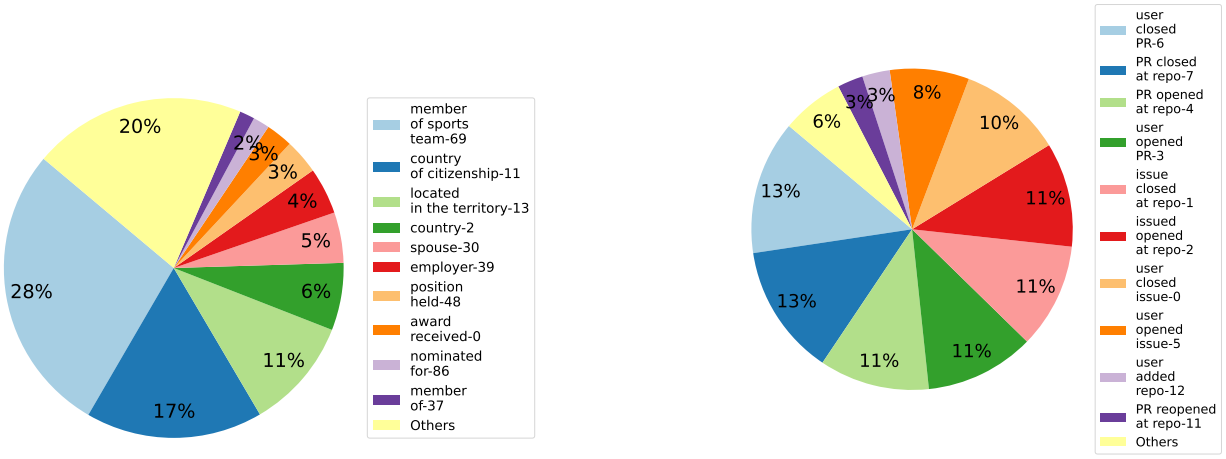


(a) `tkgl-smallpedia` **(b)** `thgl-software`

Figure 8.2: Number of edges over time.

granularity and trend patterns. TKG datasets demonstrate a coarser time granularity leading to a significantly higher edge count per timestep compared to THG datasets. `thgl-software` exhibits a relatively constant number of edges over time (with peaks at specific time points). In comparison, `tkgl-smallpedia` exhibits significant growth in edge count closer to the end. This is because `tkgl-smallpedia` starts in 1900 and ends in 2024, as time gets closer to the current era, the amount of digitized and documented information increases significantly. The reduced number of edges in the final bin is due to the fact that the knowledge from 2024 remains incomplete as of this writing.

Figure 8.3 illustrates the distribution of the ten most prominent relations in `tkgl-smallpedia` and `thgl-software`. There are highly frequent relation types in `tkgl-smallpedia` such as a member of sports team which occupies 28% of all edges; the portion of edges quickly reduces for other relations. In `thgl-software`, there is a relatively even split in the portion of edges for the most prominent relations with the top seven relations each occupying more than 10% of edges. These figures show the diversity of relations and their distributions in TGB 2.0.



(a) `tkgl-smallpedia` **(b)** `thgl-software`

Figure 8.3: Most frequent relation types for `tkgl-smallpedia` and `thgl-software` datasets. *Others* refers to all remaining relations not shown here.

8.4 Experiments

Evaluation Protocol. In TGB 2.0, we focus on the *dynamic link property prediction task* where the goal is to predict the property (often existence) of a link between a pair of nodes in a future timestamp. Here, we treat the link prediction task as a ranking problem similar to [88, 94, 65]. The model is required to assign the true edge with the highest probability from multiple negative edges (also referred to as corrupted triples in the TKG literature). The evaluation metric is the time-aware filtered Mean Reciprocal Rank (MRR) following [65, 94]. The MRR computes the average of the reciprocals of the ranks of the first relevant item in a list of results. The time-aware filtered MRR removes any edge that are known to be true at the same time as the true edge (i.e. temporal conflicts) from the list of possible destinations. For THG datasets, we predict the tails of queries $(s, r, ?, t^+)$, as in [120, 165]. Following the practice in TKG literature [65], we predict entities in both directions for TKG datasets, namely both $(s, r, ?, t^+)$ and $(?, r, o, t^+)$, achieved by introducing inverse relations where the head and tail of an existing relation is inverted. Due to the large size of TGB 2.0 datasets, we select the number of negative edges q for each dataset considering the trade-off between the evaluation completeness and the test infer-

ence time. Therefore, we utilize two negative sampling strategies for evaluation: *1-vs-all* and *1-vs-q*. For both strategies, the temporal conflicts are removed for correctness. All negative samples are then pre-generated to ensure reproducible evaluation. Lastly, any methods that uses more than 40 GB GPU memory or runs for more than a week are considered as Out Of Memory (OOM) or Out Of Time (OOT), respectively.

1-vs-all. For datasets where there is a small number of nodes, it is possible to evaluate all the possible destinations, thus achieving a comprehensive evaluation. In TGB 2.0, we use *1-vs-all* strategy for `tkgl-smallpedia`, `tkgl-polecat`, and `tkgl-icews` due to their smaller node size (see Table 8.1).

1-vs-q. For datasets with a large number of nodes, sampling q negative edges is required to achieve a practically feasible inference time. We find that randomly sampling the negative edges, and omitting the edge types, results in over-optimistic MRRs, making the prediction easy. We thus propose to incorporate the edge-type information into the negative sampling process for a more robust evaluation. For the large TKG dataset `tkgl-wikidata`, we first identify possible tails for each edge type throughout the dataset and then sample the negatives based on the edge type of the query. If there are not enough tails in a given edge type, we then randomly sample the remaining ones. For all THG datasets, we sample all destination nodes with the same node type as the true destination node, thus considering the tail node type associated with a given edge type. Through an ablation study, we found that our sampling results in closer MRR to that of the *1-vs-all* than random sampling.

8.4.1 Temporal Knowledge Graph Experiments

For TKG experiments, we include RE-GCN [124], TLogic [128], CEN [122] as well as two deterministic heuristic baselines: the Recurrency Baseline (RecB) [64] and EdgeBank [158]. For RecB, we report two versions where applicable: $\text{RecB}_{\text{default}}$ which uses default values for its two parameters, and $\text{RecB}_{\text{train}}$ which selects the optimal values for these based on performance on the validation set. For EdgeBank, we report two versions follow-

Table 8.2: MRR results for *Temporal Knowledge Graph* link property prediction task. We report the average and standard deviation across 5 different runs. First place is **bolded** and the second place is underlined.

Method	tkgl-smallpedia		tkgl-polecat		tkgl-icews		tkgl-wikidata	
	Validation	Test	Validation	Test	Validation	Test	Validation	Test
$\text{EdgeBank}_{\text{tw}}$ [158]	0.457	0.353	0.058	0.056	0.020	0.020	0.633	0.535
EdgeBank_{∞} [158]	0.401	0.333	0.048	0.045	0.008	0.009	0.632	0.535
$\text{RecB}_{\text{train}}$ [64]	**0.694**	**0.655**	0.203	0.198	OOT	OOT	OOT	OOT
$\text{RecB}_{\text{default}}$ [64]	0.640	0.570	0.170	0.167	0.264	**0.206**	OOT	OOT
RE-GCN [124]	0.631±0.001	0.594±0.001	0.191±0.003	0.175±0.002	0.232±0.003	0.182±0.003	OOM	OOM
CEN [122]	0.646±0.001	0.612±0.001	0.204±0.002	0.184±0.002	0.244±0.002	0.187±0.003	OOM	OOM
TLogic [128]	0.631±0.000	0.595±0.001	**0.236**±0.001	**0.228**±0.001	**0.287**±0.001	0.186±0.001	OOT	OOT

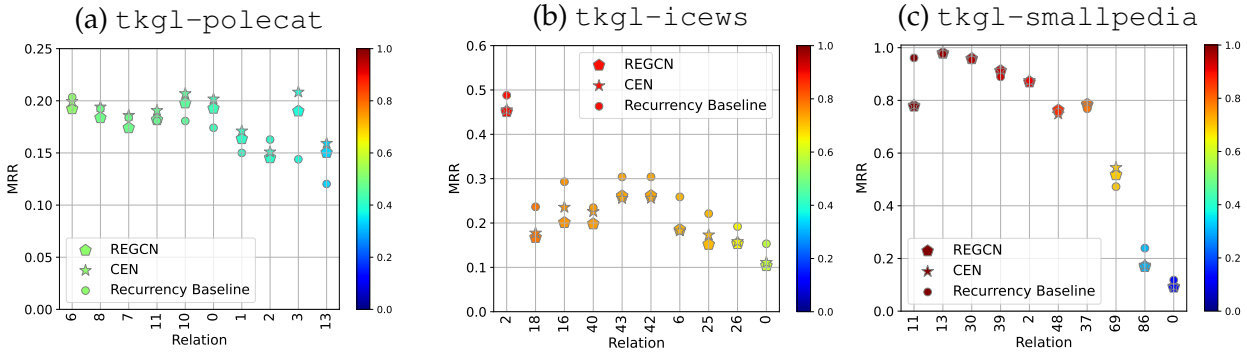


Figure 8.4: MRR per relation for the 10 highest occurring relations for three TKG datasets for RE-GCN, CEN, and Reccurency Baseline. The color indicates the Recurrency Degree value for the relation type. The relations for each dataset are ordered by decreasing Recurrency Degrees.

ing [158], $\text{EdgeBank}_{\text{tw}}$, which accounts for information from a fixed past time window, and EdgeBank_{∞}, which uses information from all past temporal triples.

We report the average performance and standard deviation across 5 runs for each method in Table 8.2. Several methods encountered out-of-memory or out-of-time errors on some datasets. The results reveals that no single method exhibits superior performance across all four datasets. Surprisingly, the RecB heuristic performs competitively across most datasets while being among the best performing on `tkgl-smallpedia` and `tkgl-icews`, underscoring the importance of including simple baselines in comparison and suggesting potential areas for improvement in other methods. The Edge-

bank heuristic, originally designed for homogenous temporal graphs, exhibits low performance, highlighting the importance of utilizing rich multi-relational information for TKG learning. On the large `tkgl-wikidata` dataset, however, Edgebank is the only method that can scale to such size, likely due to the fact that it omits edge-type information. This highlights the need for scalable methods. On another note, methods achieve higher MRRs on datasets characterized by high Recurrency Degrees and Consecutiveness values (`tkgl-smallpedia`, `tkgl-wikidata`), despite the presence of a considerable number of inductive nodes in these datasets.

Per-relation Analysis . Figure 8.4 illustrates the performance per relation of selected methods across three datasets. For each dataset, we show the ten most frequent relations, ordered by decreasing Recurrency Degree with the color reflecting the Recurrency Degree of each relation. Note that the y-axis scale varies across datasets. We observe distinct patterns in relation-specific performance across datasets: while results on `tkgl-polecat` exhibit consistent performance levels across relations, suggesting a relative homogeneity, results on the `tkgl-smallpedia` dataset show significant variance in performance, indicating a higher degree of variation among relations. Interestingly, there is a strong correlation between the Recurrency Degree and performance, most evident within the `tkgl-smallpedia` dataset.

8.4.2 Temporal Heterogenous Graph Experiments

For THG experiments, we include TGN [165] (with and without edge type information), STHN [120], RecB [64], and EdgeBank [158] for comparison. Table 8.3 reports the average performance and standard deviation across 5 runs for each method. We observe that scalability is a significant challenge for THG methods on large datasets such as `thgl-forum` and `thgl-myket`. Most methods either are out of memory or out of time for these datasets. STHN achieves the highest performance on `thgl-software` dataset showing method designed for THG can achieve significant performance gain. However, STHN

Table 8.3: MRR results for *Temporal Heterogeneous Graph Link Prediction* task. We report the average and standard deviation across 5 different runs. First place is **bolded** and the second place is underlined.

Method	thgl-software		thgl-forum		thgl-github		thgl-myket	
	Validation	Test	Validation	Test	Validation	Test	Validation	Test
EdgeBank$_{tw}$ [158]	0.279	0.288	0.534	0.534	0.355	0.374	0.248	0.245
EdgeBank$_{\infty}$ [158]	0.399	0.449	0.612	0.617	0.403	0.413	0.430	0.456
RecB$_{default}$ [64]	0.106	0.099	0.552	0.561	OOT	OOT	OOT	OOT
TGN [165]	0.299$\pm$0.012	0.324$\pm$0.017	0.598$\pm$0.086	0.649$\pm$0.097	OOM	OOM	OOM	OOM
TGN$_{edge\text{-}type}$	0.376$\pm$0.010	0.424$\pm$0.013	**0.767**$\pm$0.005	**0.729**$\pm$0.009	OOM	OOM	OOM	OOM
STHN [120]	**0.764**$\pm$0.025	**0.731**$\pm$0.005	OOM	OOM	OOM	OOM	OOM	OOM

is the least scalable, requiring 185 GB of memory for `thgl-software` to compute subgraphs, and is unable to scale to other datasets. The widely-used TGN model [165] for single-relation temporal graph learning is also adapted here, with a modification where it incorporates the edge type information as an edge feature. We observe significant improvement when TGN utilizes edge-type data, thus highlighting the potential to leverage the multi-relational information in THGs. Lastly, the EdgeBank heuristic achieves competitive performance with that of TGN while being scalable to large datasets. Therefore, it is important to evaluate against simple baselines to understand method performances.

8.5 Conclusion

In this chapter, we introduce TGB 2.0, a novel benchmark for reproducible, realistic, and robust evaluation on multi-relational temporal graphs that is building on the Temporal Graph Benchmark (TGB). We present four new TKG and four new THG datasets which introduce multi-relation datasets in TGB. TGB 2.0 datasets are significantly larger than existing ones while being diverse in statistics and dataset domains. TGB 2.0 focuses on the dynamic link property prediction task and provides an automated pipeline for dataset downloading, processing, method evaluation, and a public leaderboard to track progress. From our experiments, we find that both TKG and THG methods struggle to tackle large-

scale datasets in TGB 2.0, often resulting in overly long runtime or exceeding the memory limit. Therefore, scalability is an important future direction. Another observation is that heuristic methods achieve competitive results on TKG and THG datasets. This highlights the importance of the inclusion of simple baselines and underlines the room for improvement in current methods.

Chapter 9

UTG: Towards a Unified View of Snapshot and Event Based Models for Temporal Graphs

In Chapter 7 and 8, we proposed novel benchmarks to extensively evaluate current temporal graph learning methods. The focus of TGB was in the evaluation of TGL methods that model temporal graphs as a a stream of edge events, or CTDGs. In Chapter 4, 5 and 6, temporal graphs are also represented as a sequence of graph snapshots, or DTDGs. Until now, the development of machine learning methods for both types has occurred largely in isolation, resulting in limited experimental comparison and theoretical cross-pollination between the two. In this paper, we introduce Unified Temporal Graph (UTG), a framework that unifies snapshot-based and event-based machine learning models under a single umbrella, enabling models developed for one representation to be applied effectively to datasets of the other. We also propose a novel UTG training procedure to boost the performance of snapshot-based models in the streaming setting. We comprehensively evaluate both snapshot and event-based models across both types of temporal graphs on the temporal link prediction task. Our main findings are threefold: first, when combined with UTG training, snapshot-based models can perform competitively with event-based

models such as TGN and GraphMixer even on event datasets. Second, snapshot-based models are at least an order of magnitude faster than most event-based models during inference. Third, while event-based methods such as NAT and DyGFormer outperforms snapshot-based methods on both types of temporal graphs, this is because they leverage joint neighborhood structural features thus emphasizing the potential to incorporate these features into snapshot-based models as well. These findings highlight the importance of comparing model architectures independent of the data format and suggest the potential of combining the efficiency of snapshot-based models with the performance of event-based models in the future. This chapter is based on our work currently under review [95].

Reproducibility: The datasets for this work is publicly available on Zenodo [*]. The code for this project will be made publicly available after review period.

9.1 Introduction

Recently, Graph Neural Networks (GNNs)[107, 197] and Graph Transformers[221, 160] have achieved remarkable success in various tasks for static graphs, such as link prediction, node classification, and graph classification [88]. These successes are driven by standardized empirical comparisons across model architectures [88] and theoretical insights into the expressive power of these models [214].

However, real-world networks such as financial transaction networks [177], social networks [141], and user-item interaction networks [115] are constantly evolving and rarely static. These evolving networks are often modeled by Temporal Graphs (TGs), where entities are represented by nodes and temporal relations are represented by timestamped edges between nodes. Temporal graphs are categorized into two types: Discrete-Time Dynamic Graphs (DTDGs) and Continuous-Time Dynamic Graphs (CTDGs) [104]. DTDGs are represented by an ordered sequence of graph snapshots, while CTDGs consist

[*] https://zenodo.org/records/7213796

of timestamped edge streams. Both representations of temporal graphs are prevalent in real-world applications.

Until now, the development of ML methods for both types has occurred mostly independently, resulting in limited experimental comparison and theoretical cross-pollination between the two. We argue that the time granularity of the data collection process together with the requirements of the downstream task have created a gap between DTDG and CTDG in *model development* and *evaluation*.

Isolated Model Development. Despite the similarities between DTDGs and CTDGs, models for these graphs have been developed largely in isolation. Adopting the terminology of [131], models targeting DTDGs focus on learning from a sequence of graph snapshots (*snapshot-based models*) [218, 28, 153] , while methods for CTDGs focus on learning from a stream of timestamped edge events (*event-based models*) [165, 158, 133]. The disparate data representations of DTDGs and CTDGs have impeded comprehensive comparison across models developed for each category. Consequently, there are limited theoretical insights and empirical evaluations of the true potential of these models when compared together. In real-world applications, representing the data as CTDGs or DTDGs is often a design choice, and the ambiguity of the actual performance merits of both categories makes it challenging to select the optimal model in a practical setting.

Distinct Evaluation Settings. Another obstacle to comparing snapshot and event-based methods is their distinct evaluation settings. Snapshot-based methods have been primarily tested under the *deployed setting* [218, 153] [†], where the test set information is strictly not available to the model, and training set information is used for prediction. In contrast, event-based models are designed for the *streaming setting* [165, 94], where streaming predictions allow the model to use recently observed information, enabling event-based models to update their node representations at test time.

[†] An exception to this is ROLAND [223], which proposed the live-update setting.

In this work, we aim to bridge the gap between event-based and snapshot-based models by providing a unified framework to train and evaluate them to predict future events on any type of temporal graph. Our main contributions are as follows:

- **Unified framework**: We propose *Unified Temporal Graph* (UTG), a framework that unifies snapshot-based and event-based temporal graph models under a single umbrella, enabling models developed for one representation to be applied effectively to datasets of the other.

- **Updating snapshot-based models**: We propose a novel UTG training strategy to boost the performance of snapshot-based models in the streaming setting. This allows snapshot-based models to achieve competitive performance with event-based models such as TGN and GraphMixer on the `tgbl-wiki` and Reddit CTDG datasets.

- **Benchmarking**: By leveraging the UTG framework, we conduct the first systematic comparison between snapshot and event-based models on both CTDG and DTDG datasets. While some event-based methods such as NAT and DyGFormer outperform snapshot-based methods on both CTDGs and DTDGs, we posit this is due to leveraging joint neighborhood structural features rather than a fundamental property of event-based methods. Additionally, snapshot-based methods are at least an order of magnitude faster than event-based methods while achieving competitive performance. This suggests several future directions, such as integrating joint neighborhood structural features in snapshot-based models and developing a universal method that combines accuracy and efficiency for both DTDGs and CTDGs.

9.2 Related Work

Holme *et al.* [85] provided a general overview of the many types of real-world temporal networks showing that temporal graphs are ubiquitous in many applications. Recently, many ML methods have been developed for temporal graphs. The well-adopted catego-

rization by Kazemi *et al.* [104] are defined by the two types of temporal graphs: Discrete Time Dynamic Graphs (DTDGs) and Continuous Time Dynamic Graphs (CTDGs). Due to differences in input data format, empirical comparison between methods designed for CTDGs and DTDGs are under-explored and these two categories are often considered distinct lines of research despite many similarities in models' design.

Discrete Time Dynamic Graphs. Early methods often represent temporal graphs as a sequence of graph snapshots while adapting common graph neural networks such as Graph Covolution Network (GCN) [107] used in static graphs for DTDGs. For example, EGCN [153] employs a Recurrent Neural Network (RNN) to evolve the parameters of a GCN over time. In comparison, GCLSTM [28] learns the graph structure via a GCN while capturing temporal dependencies with an LSTM network [82]. Pytorch Geometric-Temporal (PyG Temporal) [166] is a comprehensive framework that facilitate neural spatiotemporal signal processing which implements existing work such as EGCN and GCLSTM in an efficient manner. HTGN [218] utilizes hyperbolic geometry to better capture the complex and hierarchical nature of the evolving networks. Recently, You *et al.* [223] introduced a novel *live-update setting* where GNNs are always trained on the most recent observed snapshot after making predictions. In comparison, the *streaming setting* in this work allows the model to use observed snapshots for forward pass but no training is permitted on the test set. Zhu *et al.* [236] designed the WinGNN framework for the live-update setting where a simple GNN with meta-learning strategy is used in combination with a novel random gradient aggregation scheme, removing the need for temporal encoders.

Continuous Time Dynamic Graphs. Event-based methods process temporal graphs as a stream of timestamped edges. DyRep [195] and JODIE [115] are two pioneering work on CTDGs. TGAT [213] is one of the first works for studying inductive representation learning on temporal graphs. Rossi *et al.* [165] introduce Temporal Graph Networks (TGNs), a generic inductive framework of Temporal Graph Networks, showing DyRep, JODIE and TGAT as their special cases. Methods such as CAWN [206] and

NAT [133] both focus on learning the joint neighborhood of the two nodes of interest in the link prediction task. CAWN focuses on learning from temporal random walks while NAT is a neighborhood-aware temporal network model that introduces a dictionary-type neighborhood representation for each node. TCL [205] and DyGFormer [224] apply transformer-based architecture on CTDGs, inspired by the success of transformer-based architectures on time series [209], images [46] and natural language processing [22]. Despite the promising performance of event-based methods, recent work showed significant limitations in the standard link prediction evaluation due to the simplicity of negative samples used for evaluation [158].

9.3 Preliminaries

Definition 3 (Continuous Time Dynamic Graphs)**.** *A Continuous Time Dynamic Graph (CTDG) $\mathcal{G}$ is formulated as a collection of edges represented as tuples with a source node, destination node, start time, and end time;*

$$\mathcal{G} = \{(s_0, d_0, t_0^s, t_0^e), (s_1, d_1, t_1^s, t_1^e), \ldots, (s_k, d_k, t_k^s, t_k^e)\}$$

where, for edge $i \in [0, k]$, s_i and d_i denote source and destination respectively. The start times are ordered chronologically $t_0^s \leq t_1^s \leq ... \leq t_k^s$, each start time is less than or equal to the corresponding end time $t_i^s \leq t_i^e$, hence for each timestamp t we have $t \in [t_0^s, t_k^e]$.

Without loss of generality, one can normalize the timestamps in $\mathcal{G}$ from $[t_0, t_k]$ to $[0, 1]$ by applying $t = \frac{t - t_0}{t_k} \forall t \in [t_0, t_k]$. Real-world temporal networks can be broadly classified into two inherent types based on the nature of their edges: *spontaneous networks* and *relationship networks*. Examples of spontaneous networks include transaction networks, retweet networks, Reddit networks, and other activity graphs. Here, the edges are spontaneous thus resulting in the start time and end time of an edge being the same, i.e. $t^s = t^e$. This formulation is inline with related studies in [158, 104, 94, 174]. For relationship net-

works such as friendship networks, contact networks, and collaboration networks, the edges often persist over a period of time resulting in $t^s \neq t^e$.

Definition 4 (Discrete Time Dynamic Graphs)**.** *A Discrete Time Dynamic Graph (DTDG)* $\mathbf{G}$ *is a sequence of graph snapshots sampled at regularly-spaced time intervals [104]:*

$$\mathbf{G} = \{\mathbf{G}_0, \mathbf{G}_1, \ldots, \mathbf{G}_T\}$$

$\mathbf{G}_t = \{\mathbf{V}_t, \mathbf{E}_t\}$ *is the graph at snapshot* $t \in [0, T]$*, where* $\mathbf{V}_t$*,* $\mathbf{E}_t$ *are the set of nodes and edges in* $\mathbf{G}_t$*, respectively.*

9.4 UTG Framework

In this section, we present the Unified Temporal Graph (UTG) framework which aims to unify snapshot-based and event-based temporal graph models under the same framework, enabling temporal graph models to be applied to both CTDGs and DTDGs. UTG has two key components: *input mapper* and *output mapper*. Input mapping converts the input temporal graph into the appropriate representation needed for a given method, i.e. snapshots or events. Output mapper transforms the prediction of the model to the required time granularity of the task. Figure 9.1 shows the workflow of UTG framework. UTG enables any temporal graph learning methods to be applied to any input temporal graph via *UTG input mapper* and *UTG output mapper*.

9.4.1 UTG Input Mapper

Both snapshot and event-based TG methods require a specific input data format. For snapshot-based models, discretizing CTDG data into a sequence of snapshots is required (Section 9.4.1). For event-based models, DTDG snapshots need to be converted into batches of events (Section 9.4.1).

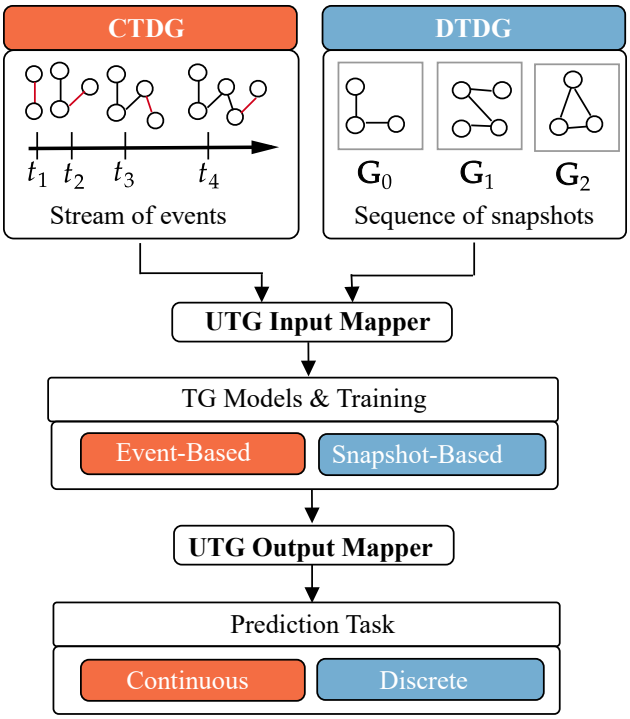


Figure 9.1: Illustration of the UTG framework. The input graph is processed by the UTG input mapper to generate the appropriate input data format for TG models. The model predictions are then processed by the UTG output mapper for prediction.

Converting CTDG to Snapshots

Here, we formulate the discretization process which converts a continuous-time dynamic graph into a sequence of graph snapshots for snapshot-based models.

Definition 5 (Discretization Partition)**.** *Let* 0 *and* 1 *be the normalized start and end time of a temporal graph* $\mathcal{G}$*. A discretization partition* $\mathbf{P}$ *of the interval* $[0, 1]$ *is a collection of intervals:*

$$\mathbf{P} = \{[\tau_0, \tau_1], [\tau_1, \tau_2], \ldots, [\tau_{k-1}, \tau_k]\}$$

such that $0 = \tau_0 < \tau_1 < \cdots < \tau_k = 1$ *and where* $k \in \mathbb{N}$*.*

Hence, a discretization partition $\mathbf{P}$ defines a finite collection of non-overlapping intervals, and its norm is defined as:

$$||\mathbf{P}|| = \max\{|\tau_1 - \tau_0|, |\tau_2 - \tau_1|, \ldots, |\tau_k - \tau_{k-1}|\}$$

The norm $||\mathbf{P}||$ can also be interpreted as the *max duration* of a snapshot in the temporal graph $\mathcal{G}$. The cardinality of $\mathbf{P}$ is denoted by $|\mathbf{P}|$.

Definition 6 (Regular Discretization Partition)**.** *A given discretization partition* $\mathbf{P}$ *is regular if and only if:*

$$\forall[\tau_i, \tau_j] \in \mathbf{P},\ |\tau_j - \tau_i| = ||\mathbf{P}|| = \frac{|\tau_k - \tau_0|}{|\mathbf{P}|}$$

In this case, all intervals have the same duration equal to $||\mathbf{P}||$.

Definition 7 (Induced Graph Snapshots)**.** *Given a Continuous Time Dynamic Graph* $\mathcal{G}$ *and a Regular Discretization Partition* $\mathbf{P}$*, the Induced Graph Snapshots* $\mathbf{G}$ *are formulated as:*

$$\mathbf{G} = \{\mathbf{G}_{\tau_0}^{\tau_1}, \mathbf{G}_{\tau_1}^{\tau_2}, \ldots, \mathbf{G}_{\tau_{k-1}}^{\tau_k}\}$$

where $\mathbf{G}_{\tau_i}^{\tau_j}$ *is defined as the aggregated graph snapshot containing all edges that have a* start time $t_s < \tau_j$ *and an* end time $t_e \geq \tau_i$*, i.e. edges that are present solely within the* $[\tau_i, \tau_j)$ *interval.*

Note that for spontaneous networks, each edge exist at a specific time point $t_s = t_e$ thus only belonging to a single interval/snapshot. For relationship networks, however, it is possible for an edge to belong to multiple intervals depending on its duration.

Definition 8 (Discretization Level)**.** *Given a regular discretization partition* $\mathbf{P}$ *and the timestamps in a temporal graph* $\mathcal{G}$ *normalized to* $[0, 1]$*, the* discretization level Δ *of* $\mathbf{P}$ *is computed as :*

$$\Delta = \frac{1}{|\mathbf{P}|}$$

where $|\mathbf{P}|$ *is the cardinality of the partition or the number of intervals.*

Note that $\Delta \in [0, 1]$. When $|\mathbf{P}| = 1$ then $\Delta = 1$ which means the temporal graph is collapsed into a single graph snapshot (i.e. a static graph). On the other extreme, we have $\lim_{|\mathbf{P}|\to\infty} \Delta = 0$, preserving the continuous nature of the continuous time dynamic graph $\mathcal{G}$.

Definition 9 (Time Gap)**.** *Given a Continuous Time Dynamic Graph $\mathcal{G}$ and a Regular Discretization Partition $\mathbf{P}$, a Time Gap occurs when there exist one or more snapshots in the Induced Graph Snapshots $\mathbf{G}$ with an empty edge set.*

In this work, we choose the number of intervals in discretization by selecting the finest time granularity which would not induce a time gap. This ensures that there are no empty snapshots in the induced graph snapshots.

Converting DTDG to Events

While it may seem straightforward to convert DTDG to events — simply create one event with timestep t for each edge in snapshot G_t — some subtleties related to batch training and memory update of event-based models have to be considered to avoid data leakage. Event-based models often receive batches of events (or edges) with a fixed dimension as inputs [165, 224, 133]. In discrete-time dynamic graphs, all edges in a snapshot have the same timestamp and are assumed to arrive simultaneously. Therefore, using a fixed batch size can result in splitting the snapshot into multiple batches. Because models in the *streaming setting* [94] such as TGN [165] and NAT [133] update their representation of the temporal graph at the end of each batch, predicting a snapshot across multiple batches leads to data leakage: a portion of the edges from the snapshot is used to predict other (simultaneous) edges from the same snapshot. To avoid data leakage on DTDGs, we ensure that each snapshot is contained in a single batch for event-based models[‡]. Note that the issue of splitting edges that share the same timestamp into multiple batches can

[‡]In case the resulting batch would not fit in memory, one can delay the memory update (and parameter updates during training) only after all edges from the current snapshot have been processed, something akin to gradient accumulation.

exist in general CTDG datasets as well, more likely for datasets with a large burst of edges at a single timestamp.

9.4.2 UTG Output Mapper

The output task on the temporal graph can be either discrete or continuous. Discrete tasks refer to predicting which edges will be present at a future snapshot (with an integer timestep). Continuous tasks refer to predicting which edges will be present for a given UNIX timestamp in the future. Snapshot-based models often omits the timestamp of the prediction as an input, implicitly assuming the prediction is for the next snapshot. Therefore, applying snapshot-based models for a continuous task requires 1). always updating the model with all the information available until the most recent observed snapshot (*test-time update*) and 2). mapping the discrete-time prediction to a continuous timestamp. We explain here how to map the prediction to a continuous space with zero-order hold.

Definition 10 (Zero-order Hold)**.** *A discrete time signal $y[i]$, $i \in \mathbb{N}$, can be converted to a continuous time signal $y(t)$, $t \in \mathbb{R}$, by broadcasting the value y[i] as a constant in the interval $[\tau_i, \tau_j]$:*

$$y(t) = y[i], \ \textit{for all} \quad \tau_i \leq t \leq \tau_j$$

where $[\tau_i, \tau_j]$ specifies the duration of the discrete signal.

By applying zero-order hold for snapshot-based models, the predictions can now be broadcasted for a period of time (specifically for the duration of a given snapshot $[\tau_i, \tau_j]$). Therefore, it is now possible to utilize snapshot-based models on continuous-time dynamic graphs. Note that often snapshot-based model are designed to predict for the immediate next snapshot and are not capable of predicting snapshots in the more distant future. Therefore, inherently their ability to predict events in the far future is limited when compared to the event-based model that explicitly takes a timestamp as input. With zero-order hold, we assume that the event to predict next is within the next snapshot to

circumvallate the aforementioned limitation of the snapshot-based model. To achieve this, we select the finest time granularity on the CTDG datasets which results in no time gap to construct snapshots.

9.4.3 Streaming Evaluation

Event-based models often evaluate with the *streaming setting* [165, 224, 206, 133]. In this setting, information from the previously observed batches of events (or graph snapshots) can be used to update the model however no information from the test set is used to train the model. In comparison, snapshot-based models are often evaluated in either the *live-update setting* [223] or the *deployed setting* [153, 218].

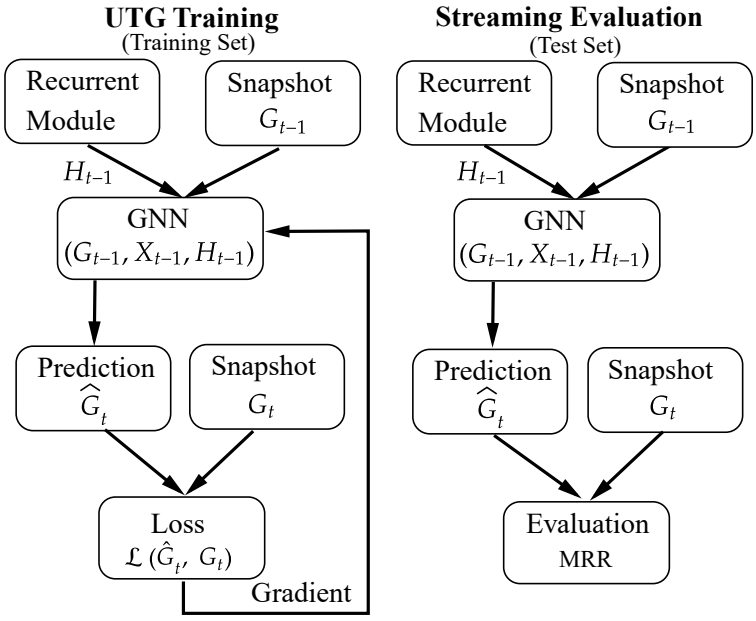


Figure 9.2: UTG training and evaluation workflow. UTG training enables snapshot-based models to incorporate information from observed snapshots at inference time.

In this work, to provide a unified comparison, we focus on the widely used streaming setting for experimental evaluation as it closely resembles real-world settings where after the model is trained, it is required to incorporate newly observed information into its predictions. Note that the UTG framework can also be applied to test under the deployed setting with little changes. For the live-update setting, the changes are required for each model's training procedure as the training set is not split chronologically but rather is a subgraph of each observed snapshot. Figure 9.2 shows the evaluation pipeline used for snapshot-based models in UTG. After a snapshot is ob-

served, it can be used to update the node representation of the snapshot-based models for the prediction of the next snapshot (only forward pass for inference).

9.4.4 UTG Training for Snapshot-based Models

Here, we discuss the changes to the snapshot-based methods in the UTG framework. Figure 9.2 illustrates the workflow of UTG training for snapshot-based models. The standard training for snapshot-based models, e.g. with Pytorch Geometric Temporal [167], are designed for tasks such as graph regression or node classification, for the deployed setting. Therefore, the training procedure needs to be adapted accordingly for the link prediction with the streaming setting.

Difference of UTG Training. The first required change is that in standard training, the snapshots up until time t are used as input for predictions at time t. This is feasible for node classification and graph regression tasks: the graph structure at time t is used to predict the unknown target labels. However, this is problematic for the link prediction task as the target itself (the graph structure) is not available as input to the model. Therefore, to account for this, we modified the training to use only snapshots up to $\mathbf{G}_{t-1}$ as input to predict the graph structure at the current step $\mathbf{G}_t$.

The second change is that, in UTG, the loss is backpropagated to the model at each snapshot. This is in contrast with accumulating the loss through the whole sequence and only backpropagation once at the end of training (as seen in standard training). This change is motivated by the training procedure often seen in event-based models for the streaming setting. For example, in TGN [165], loss on each batch is computed based on information from the previous batch. UTG training enhances the performance of snapshot-based models for the streaming setting and in Section 9.5, we demonstrate the performance advantage of UTG training on a number of snapshot-based models.

Connection to Truncated Backprop Through Time. By considering the temporal graph as a sequence (of snapshots), the link prediction problem can be seen as a time series prediction task where the information until time $(t-1)$ is used to predict for time

Table 9.1: Dataset statistics.

	Dataset	# Nodes	# Edges	# Unique Edges	Surprise	Time Granularity	# Snapshots
DTDG	UCI	1,899	26,628	20,296	0.535	Weekly	29
	Enron	184	10,472	3,125	0.253	Monthly	45
	Contact	694	463,558	79,531	0.098	Hourly	673
	Social Evo.	74	87,479	4,486	0.005	Daily	244
	MOOC	7,144	236,808	178,443	0.718	Daily	31
CTDG	`tgbl-wiki`	9,227	157,474	18,257	0.108	UNIX timestamp	745 (Hourly)
	`tgbl-review`	352,637	4,873,540	4,730,223	0.987	UNIX timestamp	237 (Monthly)
	Reddit	10,984	672,447	78,516	0.069	UNIX timestamp	745 (Hourly)

t. Many snapshot-based models utilize a RNN to model temporal dependency. In this view, accumulating gradients throughout the whole sequence before backpropagation is equivalent to the classical backpropagation through time algorithm [210]. From this perspective, backpropagating the loss at each snapshot in UTG training can be interpreted as using the Truncated Backpropagation Through Time (TBTT) algorithm to train the model, with a window size of one. It is known that TBTT helps circumvent common issues of training RNNs such as exploding memory usage and vanishing gradient problem [9, 154]. Therefore, UTG training might help alleviate the vanishing gradient problem.

9.5 Experiments

In this section, we benchmark both snapshot-based and event-based methods across both CTDG and DTDG datasets under the UTG framework.

Datasets. In this work, we consider five discrete-time dynamic graph datasets and three continuous-time dynamic graph datasets. `tgbl-wiki` and `tgbl-review` are datasets from TGB [94] while the rest are found in [158]. The dataset statistics are shown in Table 9.1. The time granularity or discretization level of each DTDG dataset is selected as the finest time granularity where there are no time gaps (see Section 9.4.1). The surprise index is defined as $surprise = \frac{|E_{\text{test}} \setminus E_{\text{train}}|}{E_{\text{test}}}$ [158] which measures the proportion of unseen edges in the test set when compared to the training set.

Evaluation Setting. A common approach for evaluating dynamic link prediction tasks is similar to binary classification, where one negative edge is randomly sampled for each positive edge in the test set, and performance is measured using metrics like the Area Under the Receiver Operating Characteristic curve (AUROC) or Average Precision (AP) [165, 213]. However, recent studies have shown that such evaluation is overly simplistic and tends to inflate performance metrics for most models [158, 94]. One main reason is that randomly sampled negative edges are too easy and the more challenging *historical negatives* (past edges absent in the current timestamp) are rarely sampled [158]. To address these issues, several improvements have been proposed, framing the problem as a ranking task where the model must identify the most probable edge from a large pool of negative samples as well as adding challenging negatives. Therefore, we adopt the improved evaluation methodology used in TGB [94], where link prediction is treated as a ranking problem and the Mean Reciprocal Rank (MRR) metric is applied. This metric calculates the reciprocal rank of the true destination node among a large number of possible destinations.

For each dataset, we generate a fixed set of negative samples for each positive edge consisting of 50% *historical negatives* and 50% *random negative*, same as in [94]. For DTDG datasets, we generate 1000 negative samples per positive edge. For TGB datasets, we use the same set of negatives provided in TGB and for Reddit, we generate 1000 negatives similar to before. For graphs with less than 1k nodes, we generate negative samples equal to the number of nodes. We follow the *streaming setting* where the models are allowed to update their representation at test time while gradient updates are not permitted. For DTDG datasets, we select the best results from learning rate 0.001 or 0.0002. For CTDG datasets, we report the results from TGB [94] where available or by learning rate 0.0002.

Compared Methods. We compare four event-based methods including TGN [165], DyGFormer [224], NAT [133] and GraphMixer [37]. We also include Edgebank [158], a scalable and non-parametric heuristics. In addition, we compare three existing snapshot-based methods including HTGN [218], GCLSTM [28], EGCNo [153] and ROLAND-GRU [223].

Table 9.2: Test MRR comparison for snapshot and event-based methods on DTDG datasets, results reported from 5 runs. The top three models are marked by **First**, Second, *Third*.

	Method	UCI	Enron	Contacts	Social Evo.	MOOC
event	TGN [165]	0.091 ± 0.002	0.191 ± 0.027	0.153 ± 0.007	0.283 ± 0.009	0.174 ± 0.009
	DyGFormer [224]	0.334 ± 0.024	**0.331** ± 0.010	**0.283** ± 0.006	**0.366** ± 0.004	OOM
	NAT [133]	**0.356** ± 0.048	*0.276* ± 0.014	0.245 ± 0.015	0.258 ± 0.036	**0.283** ± 0.058
	GraphMixer [37]	0.105 ± 0.008	0.296 ± 0.019	0.055 ± 0.003	0.157 ± 0.005	OOM
	EdgeBank$_{\infty}$ [158]	0.055	0.115	0.016	0.049	0.040
	EdgeBank$_{tw}$ [158]	*0.165*	0.157	0.050	0.070	0.070
snapshot	HTGN (UTG) [218]	0.093 ± 0.012	0.267 ± 0.007	0.165 ± 0.001	0.228 ± 0.003	0.093 ± 0.005
	GCLSTM (UTG) [28]	0.093 ± 0.006	0.170 ± 0.008	0.128 ± 0.004	*0.286* ± 0.003	*0.143* ± 0.006
	EGCNo (UTG) [153]	0.121 ± 0.010	0.233 ± 0.008	*0.192* ± 0.001	0.253 ± 0.006	0.126 ± 0.009
	GCN (UTG) [107]	0.068 ± 0.009	0.164 ± 0.011	0.104 ± 0.002	0.289 ± 0.008	0.084 ± 0.010
	ROLAND (UTG) [223]	0.103 ± 0.011	0.243 ± 0.017	0.145 ± 0.002	0.240 ± 0.005	0.121 ± 0.003

Lastly, we adopt a common 2-layer (static) GCN [107] under the UTG framework to demonstrate the flexibility of UTG (without a recurrent module). If a method runs out of memory on a NVIDIA A100 GPU (40GB memory), it is reported as out of memory (OOM). If a method runs for more than 5 days, it is reported as out of time (OOT).

9.5.1 Comparing Event-Based with Snapshot-Based Models

With the UTG framework, we can now compare snapshot-based methods and event-based methods on any temporal graph dataset. This comparison allows us to focus on analyzing the strengths and weaknesses of the model design, independent of the data format.

DTDG Results. Table 9.2 shows the performance of all methods on the DTDG datasets. Surprisingly, we find that event-based methods achieve state-of-the-art performance on the DTDG datasets, particularly with DyGFormer and NAT consistently outperforming other methods.

Table 9.3: Test MRR comparison for snapshot and event-based methods on CTDG datasets, results reported from 5 runs. Top three models are colored by **First**, Second, *Third*.

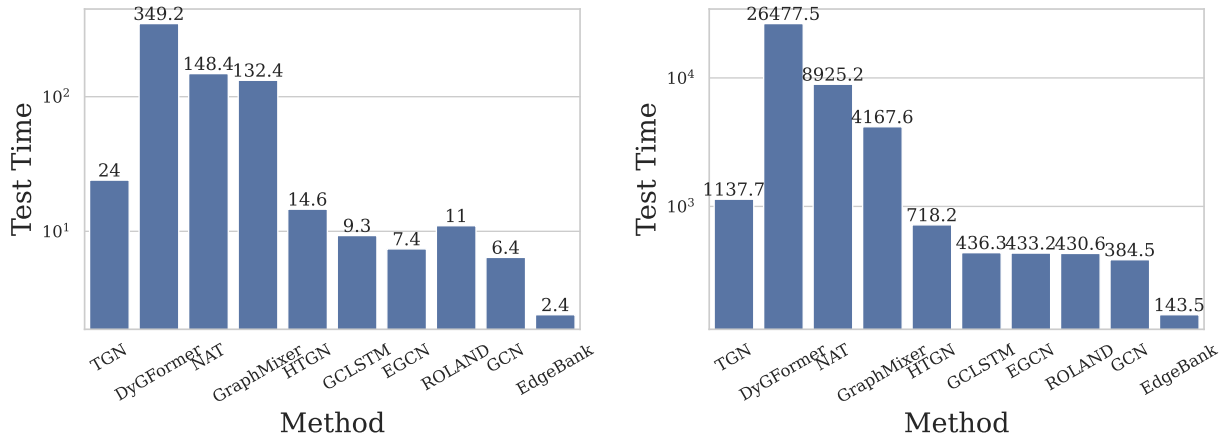
	Method	tgbl-wiki	tgbl-review	Reddit
event	TGN [165]	0.396 ± 0.060	0.349 ± 0.020	0.499 ± 0.011
	DyGFormer [224]	**0.798** ± 0.004	0.224 ± 0.015	OOT
	NAT [133]	0.749 ± 0.010	*0.341* ± 0.020	**0.693** ± 0.015
	GraphMixer [37]	0.118 ± 0.002	**0.521** ± 0.015	0.136 ± 0.078
	EdgeBank$_{\infty}$ [158]	0.495	0.025	0.485
	EdgeBank$_{tw}$ [158]	*0.571*	0.023	0.589
snapshot	HTGN (UTG) [218]	0.464 ± 0.005	0.104 ± 0.002	*0.533* ± 0.007
	GCLSTM (UTG) [28]	0.374 ± 0.010	0.095 ± 0.002	0.467 ± 0.004
	EGCNo (UTG) [153]	0.398 ± 0.007	0.195 ± 0.001	0.321 ± 0.009
	GCN (UTG) [107]	0.336 ± 0.009	0.186 ± 0.002	0.242 ± 0.005
	ROLAND (UTG) [223]	0.289 ± 0.003	0.297 ± 0.006	0.211 ± 0.006

With the improvements from UTG, snapshot-based models can obtain competitive performance on datasets such as Enron and Social Evo. Interestingly, even the simple GCN with UTG training can achieve second place performance on the Social Evo. dataset. Note that this dataset has the lowest surprise out of all datasets meaning the majority of test set edges have been observed during training, possibly explaining the strong performance of GCN in this case. Lastly, on the MOOC dataset which has the largest number of nodes out of all DTDG datasets, both DyGformer and GraphMixer ran out of memory (OOM) showing their difficulty in scaling with the number of nodes in a snapshot.

CTDG Results. Table 9.3 shows the performance of all methods on the CTDG datasets. Similar to DTDG datasets, DyGformer and NAT retain competitive performance here. On the `tgbl-wiki` and Reddit dataset, HTGN, a snapshot-based model is able to outperform the widely-used TGN and GraphMixer models. This shows that learning from the discretized snapshots can be effective even on CTDG datasets. However, snapshot-based models have lower performance on the `tgbl-review` dataset where the surprise index

is high. This shows that the inductive reasoning capability on snapshot-based models should be further improved to generalize to unseen edges.

9.5.2 Computational Time Comparison



(a) Test time (sec) for Social Evo. dataset.

(b) Test time (sec) for `tgbl-review` dataset.

Figure 9.3: Snapshot-based models with UTG training are at least an order of magnitude faster than event-based models for inference.

Figure 9.3a and Figure 9.3b show the test inference time comparison for all methods on the Social Evo. and the `tgbl-review` dataset respectively. Overall, we observe that snapshot-based methods are at least an order of magnitude faster than most event-based methods. In comparison, high-performing models such as DyGformer have significantly higher computational time thus limiting its scalability to large datasets. One important future direction is to combine the predictive power of event-based models such as NAT and DyGFormer with that of the efficiency of the snapshot-based model for improved scalability.

9.5.3 Discussion

Event-based methods such as NAT and DyGFormer tend to perform best on both CTDG and DTDG, potentially leading to the premature conclusion that event-based modeling

Table 9.4: Base models with UTG training when compared to original training, results averaged across five runs, best results for each model are **bolded**.

Dataset	GCLSTM (UTG)	GCLSTM (original)	EGCNo (UTG)	EGCNo (original)	HTGN (UTG)	HTGN (original)
UCI	**0.093** ± 0.006	0.047 ± 0.006	**0.121** ± 0.010	**0.124** ± 0.007	**0.093** ± 0.012	0.052 ± 0.006
Enron	**0.170** ± 0.006	0.131 ± 0.003	**0.233** ± 0.008	0.227 ± 0.012	**0.267** ± 0.007	0.196 ± 0.025
Contacts	**0.128** ± 0.004	0.101 ± 0.031	0.192 ± 0.001	**0.195** ± 0.001	**0.165** ± 0.001	0.129 ± 0.020
Social Evo.	**0.286** ± 0.003	**0.287** ± 0.009	**0.253** ± 0.006	0.231 ± 0.009	**0.228** ± 0.003	0.178 ± 0.024
MOOC	**0.143** ± 0.006	0.076 ± 0.003	**0.126** ± 0.009	0.119 ± 0.009	**0.093** ± 0.005	0.079 ± 0.011

is the preferred paradigm and that snapshot-based models should be avoided. However, the superior performance of NAT and DyGFormer could be primarily due to their ability to leverage joint neighborhood structural features, specifically the common neighbors between the source and destination nodes of a link [133, 224]. This approach has been shown to be fundamental for achieving competitive link prediction on static graphs [229, 230]. In contrast, none of the existing snapshot-based methods incorporate joint neighborhood structural features. Therefore, the performance difference could be mainly attributed to this factor rather than an intrinsic difference between event-based and snapshot-based models. This is confirmed by the fact that event-based models such as TGN and GraphMixer, which omit these features, have no clear performance advantage over snapshot-based methods. Moreover, snapshot-based methods are more computationally efficient and might be preferred when efficiency is important. These considerations suggest that both event-based and snapshot-based methods have their own merit. We believe that combining the strengths of both approaches is an important future direction.

9.5.4 The Benefit of UTG Training

UTG training enables snapshot-based models to effectively incorporate novel information during inference. Table 9.4 shows the performance benefit of using UTG training on the GCLSTM, HTGN and EGCNo models when compared to the original training scheme

(such as in Pytorch Geometric Temporal) for DTDG datasets. UTG training significantly enhances model performance across most datasets and performs identically on the rest.

9.6 Conclusion

In this chapter, we introduce the UTG framework, unifying both snapshot-based and event-based temporal graph models under a single umbrella. With the UTG input mapper and UTG output mapper, temporal graph models developed for one representation can be applied effectively to datasets of the other. To compare both types of methods in the streaming setting for evaluation, we propose the UTG training to boost the performance of snapshot-based models. Extensive experiments on five DTDG datasets and three CTDG datasets are conducted to comprehensively compare snapshot and event-based methods. We find that top performing models on both types of datasets leverage joint neighborhood structural features such as the number of common neighbors between the source and destination node of a link. In addition, snapshot-based models can achieve competitive performance to event-based model such as TGN and GraphMixer while being an order of magnitude faster in inference time. Thus, an important future direction is to combine the strength of both types of methods to achieve high-performing and scalable temporal graph learning methods.

Chapter 10

Discussion and Conclusion

The ubiquity of evolving networks in the modern age along with the increasing availability of large-scale network data present excellent opportunities to design novel machine learning systems to extract patterns from such rich data. Temporal graph learning methods are deployed in an increasing number of applications in our daily lives. For example, for the LinkedIn social network, a large-scale graph neural network utilizing temporal graph information is deployed for the out-of-network content recommendation task [17]. Another example is the integration of dynamic networks for COVID-19 modeling where large-scale mobility networks are derived from mobile phone data and used jointly with the classical epidemic modeling approach, SEIR [26]. In this thesis, we design efficient, effective, and scalable ML approaches for temporal graphs as well as provide benchmarks to facilitate realistic evaluation of such methods thus closing the gap between current ML methods and the desired criteria for real-world applications. In Section 10.1, we discuss the limitations of presented approaches in this work. As the concluding section of this thesis, we discuss promising future directions following this thesis in Section 10.2.

In the first part of the thesis, we focus on designing novel change point detection methods that are applicable for real world dynamic graphs. Our first method LAD has proven to be an effective tool for this problem by using the Laplacian spectrum to capture the graph evolution over time. However, LAD requires computing the SVD which is

expensive for networks with more than thousands of nodes. To address the limitation on scalability, we propose SCPD which efficiently approximates the spectral density of the graph Laplacian while avoiding the computation of singular values. This allows SCPD to scale to large networks often seen in real applications. Lastly, we tackle multi-view dynamic graphs by introducing multiLAD to effectively aggregate each view.

Change point detection is an inherently unsupervised task and generally assumes a lack of curated training labels therefore our proposed methods are unsupervised and have no trainable parameters. In particular, LAD, SCPD, and multiLAD directly utilize properties computed from the graph Laplacian matrix of each snapshot to capture the change in graph structure thus avoiding the need for tuning a large number of training hyperparameters. In addition, these methods can also be deployed in an online fashion. For the prediction at each time step, our proposed methods only requires the current snapshot as well as the signature vectors for past snapshots within the context windows as input thus avoiding the need to store all past observed snapshots. Empirically on traffic networks, transaction networks, social networks, political networks and citation networks, our methods have detected anomalous changes in the temporal graph structure corresponding to significant real-world events. Thus, we believe it is promising to deploy them in practice to monitor structural changes online.

In the second part of the thesis, we dive into the evaluation of ML methods for temporal graph learning. Specifically, we aim to address limitations in existing evaluation which resulted in the over-optimistic performance of current models. To this end, we propose the Temporal Graph Benchmark (TGB), a collection of challenging and diverse benchmark datasets for realistic, reproducible, and robust evaluation for machine learning on temporal graphs. TGB improves the evaluation of temporal graph learning in both dataset selection and evaluation protocols while covering both edge and node-level tasks. As a Python package, TGB automates the process of dataset downloading and processing as well as evaluation protocols, enabling reproducible evaluation and easy comparison between methods.

As the research for evaluation of TGL methods is an ongoing research topic, TGB is a community-driven project where we will continue to incorporate community feedback for future improvement. After announcing TGB, a significant number of the TG community reached out to request the addition of more graph types in TGB. Therefore, we expand upon TGB by introducing Temporal Graph Benchmark 2.0, a novel benchmarking framework designed for evaluation of link tasks on Temporal Knowledge Graphs and Temporal Heterogeneous Graphs with a focus on large-scale datasets. Both TGB and TGB 2.0 include datasets orders of magnitude larger than existing ones while providing standardized and challenging evaluation protocols. These projects highlight the significant need for more scalable temporal graph learning methods for large-scale applications.

We are also delighted to see the positive reception and adoption of TGB from the community. On one hand, many follow-up work adopted the TGB code base as a starting point as TGB provides a comprehensive benchmark and implementation of state-of-the-art event-based methods. For example, Poštuvan *et al.* [157] benchmarked temporal graph methods for the link level anomaly detection task, adapting from the TGB implementation. On the other hand, submissions to the TGB leaderboard help the research community to track the progress on current state-of-the-art methods [232, 224]. We discuss future directions to further build upon TGB in Section 10.2.2.

In the last project of this thesis, we observe that snapshot-based and event-based methods are often treated as disjoint research areas despite being closely related. Therefore, it is important to propose a novel framework that allows fair and direct comparison between them while identifying the advantages and disadvantages of each type of method when compared with each other. To this end, we proposed the UTG framework to unify both types of models under a single umbrella, enabling models developed for one representation to be tested and deployed for another. Through the UTG framework, we conduct the first systematic analysis between snapshot-based and event-based methods on both DTDG and CTDG datasets. We find that certain methods such as NAT and DyGFormer perform competitively on both DTDG and CTDG datasets while DTDG meth-

ods can achieve competitive performance with significantly less compute time. The UTG comparison opens up many future directions and can be explored further in future work.

10.1 Limitations

This thesis provides a solid foundation towards efficient, effective and scalable learning methods on temporal graphs. However, there are limitations not addressed within the scope of this thesis which will be outlined in this section.

Limitations of Change Point Detection Methods. For our proposed change point detection methods such as LAD, SCPD and MultiLAD, the selection of the sizes for sliding windows is important. Choosing different window sizes can lead to varying results. In this work, we select the window sizes based on real-world time granularities such as weekly or annually. For practitioners, it would be important to choose the window sizes based on the application and the expected context length required for the task. In addition, both LAD and MultiLAD omit the rich node features often present in temporal graphs while SCPD utilizes separate graph summaries for structural and attribute anomalies. TGNNs such as TGN [165] and NAT [133] can directly incorporate node and edge features for prediction tasks in supervised tasks. Therefore, one promising direction to explore is how can we design unsupervised or self-supervised approaches for change point detection which integrate node features directly into the architecture.

Follow-up work on the LAD line of research also aims to address the limitations of our proposed method. For example, the LAD workflow lacks the hypothesis testing step often observed in other change point detection methods. The idea is to test one hypothesis where a change point has not occurred versus another one where a change point has indeed happened. To address this limitation, Luo *et al.* [132] define a metric space for the graph Laplacians while deriving a closed-form formula for Fréchet mean and variance under this metric. In this way, the hypothesis test can be conducted with the Fréchet

distance while estimating both the location and number of change points. In LAD, we utilize the largest eigenvalues of the graph Laplacian while the effect of using the smallest eigenvalues is not evaluated. Nagao *et al.* [142] investigated the effect of using the smallest eigenvalues for change point detection and found that using these eigenvalues can also achieve strong performance.

Limitations of TGB. The Temporal Graph Benchmark is built to be a challenging benchmark covering a diverse range of tasks and modalities seen in temporal networks. Chapter 7 provides datasets and tasks for homogeneous temporal graphs while Chapter 8 adds datasets and tasks for Temporal Knowledge Graphs (TKGs) and Temporal Heterogeneous Graphs (THGs). Due to time constraints and a lack of public data sources, node-level tasks were not included for TKGs and THGs. In addition, there exists many other types of temporal graphs not yet supported in TGB for example Spatio-temporal graphs [35, 34], brain activity networks [106, 8] and human contact networks [119, 182]. TGB will continue to expand the range of datasets and tasks while listening to community feedback and suggestions. Lastly, graph-level tasks are also currently omitted from TGB as this area is relatively less explored in the literature.

Limitations of UTG framework. Currently, we have only explored the UTG framework with existing methods for the link prediction task, it would also be interesting to explore node-level tasks and graph-level tasks in the future. In addition, more recent methods can be added in the comparison for UTG for both snapshot-based and event-based approaches such as DyGMamba [44], PopTrack [39], and more. Lastly, the UTG framework enables the design of temporal graph architectures which are independent of the expected input and output format thus opening novel directions in designing input-agnostic architectures for temporal graphs. In this work, we only explored the application of a simple GCN layer in the UTG framework while it is also possible to test more advanced layers and architectures from the static graph literature.

10.2 Future Directions

In this section, we present promising future directions for temporal graph learning and how to extend the work from this thesis.

10.2.1 Foundation Model for Temporal Graphs

The field of temporal graph learning has seen significant growth in recent years, driven by novel architectures based on graph neural networks as well as transformers [165, 232, 224, 44]. However, the majority of current research focus on addressing a single machine learning task while training and testing on one temporal graph. This means testing on a new temporal graph requires training a new model from scratch, resulting in significant computational overhead as well as expertise in selecting the correct hyperparameters.

Recently, foundation models have made significant impact across various fields including natural language processing [23, 22], computer vision [159], knowledge graph learning [59], (static) graph learning [212, 233] and molecular learning [110, 7]. An ideal foundation model is pre-trained on a diverse range of datasets and has the ability to transfer to many domains and tasks either directly or via fine-tuning. The motivation is to learn the foundations of a given field via the vast amounts of training data which then enables the foundation model to predict for unseen distributions and tasks.

Designing foundation models for temporal graphs is challenging, requiring the model to learn the graph evolution as well as how to transfer knowledge across networks. Xia *et al.* [212] outlined several challenges of building graph foundation models including 1). *Structure Heterogenity*: graph structures across domains vary significantly, 2). *Feature Heterogenity*: features on graphs can vary across networks, 3). *Fast Adaptation*: how to quickly adapt to unseen networks? and 4). *Scaling Law Emergence*: the model performance should scale favorably with the amount of data and model parameters. For temporal graphs, a fifth challenge arises: 5). *Evolution Heterogenity*: the evolution of each temporal network might be distinct and the model needs to learn how the structure patterns are

evolving over time. The research question then becomes whether the model is able to learn common evolution patterns across networks and then apply it to unseen future networks. In Figure 10.1, we illustrate these challenges of building a foundation model for temporal graphs. Specifically, challenge 1,2,4,5 would be present in the pretraining phase while challenge 3 needs to be solved when deploying the foundation model on an unseen test network.

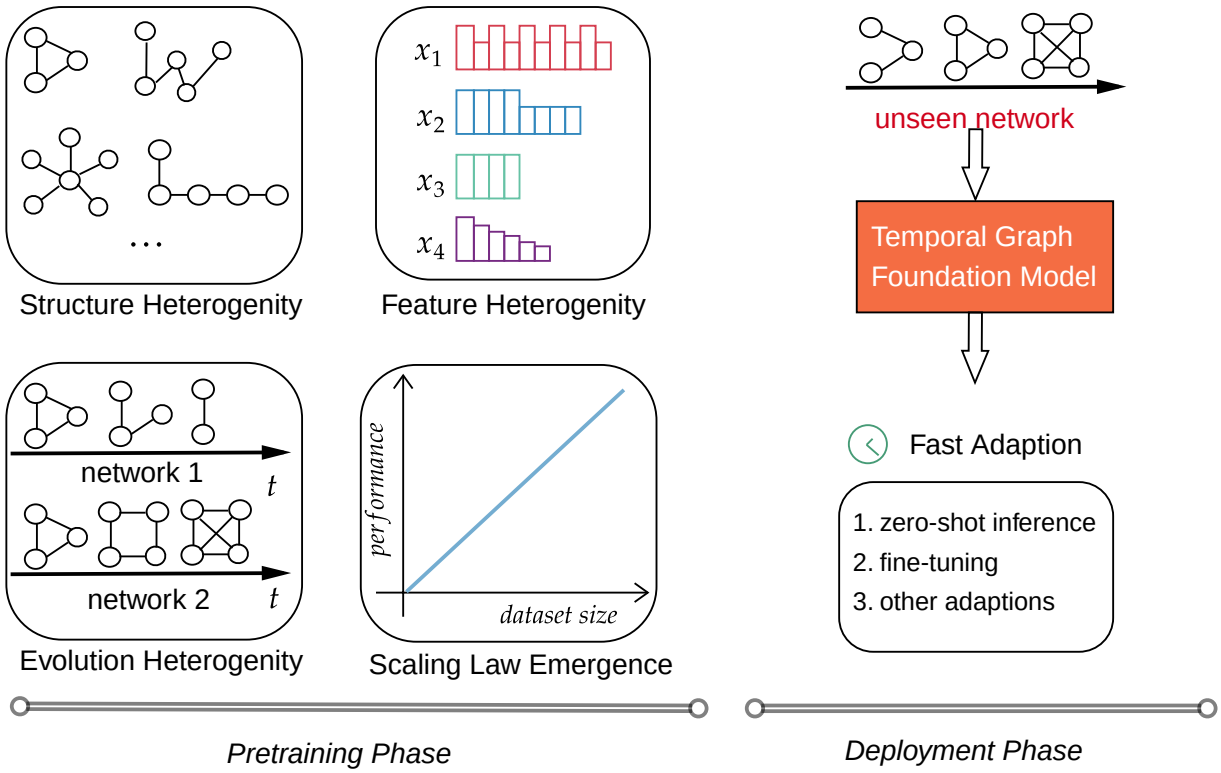


Figure 10.1: Foundation models for temporal graphs need to solve the *structural heterogenity*, *feature heterogenity*, *evolution heterogenity* and *scaling law emergency* challenges in the pretraining phase and the *fast adaptation* challenge in the deployment phase.

In a recent collaboration [183], we aim to provide some preliminary answers to some of the challenges above. In this work, we present the Temporal Graph Scaling (TGS) dataset, a collection of 84 token transaction networks collected from 2017 to 2023. We then designed the TGS training scheme which allows existing models to be trained on

multiple temporal graphs. By training the HTGN [218] model with TGS training on multiple temporal graphs for the temporal graph property prediction task, we observe that 1). it is able to generalize directly to unseen test networks within the same domain via a single inference pass, addressing *evolution heterogenity* and *fast adaptation* and 2). by training on more temporal graph networks, its performance increases significantly and surpasses the optimal performance of models trained on the test network from scratch, partially addressing *scaling law emergence*. Therefore, this is a promising first step towards building foundation models for temporal graphs. Future work can include more domain diversity in the training dataset to address *structure heterogenity* while adapting techniques such as Mixture-of-Experts can be used to adapt to *feature heterogenity* [212].

Lastly, the TGB benchmark is an excellent test bed for future foundation models. The diverse domain and scale of TGB datasets help test how well a foundation model can tackle both structure and feature heterogenity. Both node and link level tasks measures how well the foundation model is able to generalize to tasks at different levels of the graph. Lastly, the TGB leaderboard is constantly updated to reflect the current state-of-the-art, providing directly comparison to the best single model approaches.

10.2.2 Expanding Temporal Graph Benchmark

The motivation for TGB is to construct a diverse, challenging and comprehensive benchmark for tasks on temporal graphs. The TGB project is also a community-driven project where we will continue to improve the benchmark based on community suggestions. We illustrate a roadmap for TGB in Figure 10.2 where currently implemented features are marked in orange and potential future inclusions are marked in blue. Future additions to TGB can be categorized into the following: *novel settings* and *novel tasks*.

Novel Settings. Current evaluation in TGB focuses on the *streaming setting* [165, 206]. This setting is common for applications where after the model is deployed, newly arrived data should be incorporated into future predictions while updating the weights of the

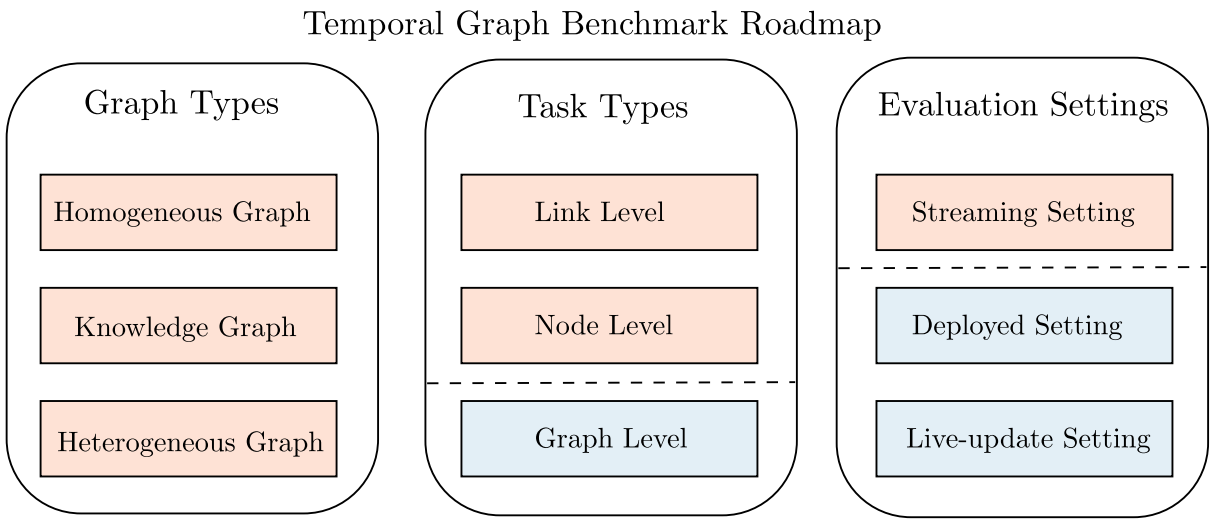


Figure 10.2: A Roadmap for extending TGB in the future. Currently implemented features are marked in orange while potential future additions are marked in blue.

model is prohibited. Other evaluation settings in the literature include the *deployed setting* [218, 153] and the *live-update setting* [223, 236]. In the deployed setting, models are prohibited from accessing any test information while in the live-update setting, models are allowed to perform gradient updates for each newly observed snapshot. For some applications, these two applications might be more appropriate. Therefore, it would be beneficial to adapt these settings for TGB datasets.

Novel Tasks. A comprehensive graph benchmark includes tasks at different levels of the graph, i.e. node, link, or graph level as well as tasks for distinct types of graphs [88, 48, 7]. A diverse range of tasks ensures that the models can provide meaningful embeddings for different entities on graphs. To start, we would like to add node-level tasks for temporal heterogeneous graph datasets such as the node affinity prediction task or the node classification task. This addition enables THG methods to be benchmarked on the less-explored node-level tasks. Recently, there is a growing interest in the temporal edge regression task on dynamic graphs [147]. Real-world tasks such as forecasting the

agriculture trade value between nations [101] and predicting the user review scores of a movie are naturally represented as the temporal edge regression task [118]. In addition, Jiang *et al.* [101] showed that TGL methods are outperformed by simple baselines such as persistence forecast and historical average in this task. Therefore, it is important to include this task in TGB to conduct a comprehensive evaluation of current TGL methods and identify future directions for improvement.

Another interesting direction is to introduce graph-level tasks into TGB. One option is the temporal graph classification task where the model is asked to predict the label of a temporal graph [61]. This is similar to the graph classification task in static graph learning where a single label is assigned for a given network [88]. In comparison, Shamsi *et al.* [176] introduced the temporal graph property prediction task where the objective is to predict the structural property of a temporal graph in the future. The focus of this task is to learn the evolution of the temporal graph property over time and leverage this knowledge to forecast it for the future. Both of these tasks will test how well a TGL method can generate graph-level embeddings, currently not measured in TGB.

10.3 Concluding Remark

Temporal graph learning is a rapidly growing field, attracting newcomers from graph learning, time series analysis and other communities. Success stories of TG applications also become more prevalent. To design practical and robust methods for the abundant opportunities for TG applications, it is important to conduct fundamental research for temporal graph learning including evaluation, theory, explainability, adversarial robustness, and more.

Recently, a large number of novel architectures have been proposed for temporal graph learning. For example, state space model-based architecture is applied for CTDGs for their high efficiency in terms of computational resources and effectiveness in capturing long-term dependency [121, 44] and models using hash tabled representations to keep

track of historical interactions [134, 133]. However, there lacks a unified expressiveness framework to understand the representation power of each category of methods along with their efficiency and effectiveness trade-offs. The temporal graph benchmark proposed in this thesis is a first step towards direct empirical comparison of these methods yet more theoretical foundations are needed to provide an understanding of the empirical results.

Another aspect explored in this work is the unification of snapshot-based and event-based methods. Our proposed UTG framework is only the first step towards this objective. Here, we focus on comparing the empirical performance of both types of methods for the streaming evaluation setting. The framework can be easily extended to other evaluation settings such as the live-update setting. In addition, the results from the UTG comparison suggest that combining the efficiency of snapshot-based methods with the joint neighborhood features often used in event-based methods can provide novel approaches that are both accurate and efficient.

Finally, as discussed in Section 10.2, there is immense potential for building foundation models in temporal graphs. With the recent success of foundation models in natural language processing, computer vision and static graphs, a promising direction is to design a multi-modal temporal graph model that jointly models the graph structure, text feature as well as image or video feature at the node/edge level. In this way, it might be possible to build machine learning models that are able to leverage complex historical interactions to predict future events and trends while describing it in human-readable format such as text or images.

Bibliography

[1] Michal Aharon, Michael Elad, and Alfred Bruckstein. K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on signal processing*, 54(11):4311–4322, 2006.

[2] Kian Ahrabian, Daniel Tarlow, Hehuimin Cheng, and Jin LC Guo. Software engineering event modeling using relative time in temporal knowledge graphs. *arXiv preprint arXiv:2007.01231*, 2020.

[3] Leman Akoglu and Christos Faloutsos. Event detection in time series of mobile communication graphs. In *Army science conference*, volume 1, 2010.

[4] Leman Akoglu, Hanghang Tong, and Danai Koutra. Graph based anomaly detection and description: a survey. *Data mining and knowledge discovery*, 29(3):626–688, 2015.

[5] Albert-László Barabási. Network science. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1987):20120375, 2013.

[6] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

[7] Dominique Beaini, Shenyang Huang, Joao Alex Cunha, Zhiyi Li, Gabriela Moisescu-Pareja, Oleksandr Dymov, Samuel Maddrell-Mander, Callum McLean, Frederik Wenkel, Luis Müller, et al. Towards foundational models for molecular

learning on large-scale multi-task datasets. In *The Twelfth International Conference on Learning Representations*.

[8] Ali Behrouz and Farnoosh Hashemi. Learning temporal higher-order patterns to detect anomalous brain activity. In *Machine Learning for Health (ML4H)*, pages 39–51. PMLR, 2023.

[9] Yoshua Bengio, Paolo Frasconi, and Patrice Simard. The problem of learning long-term dependencies in recurrent networks. In *IEEE international conference on neural networks*, pages 1183–1188. IEEE, 1993.

[10] N Benjamin Erichson, Steven L Brunton, and J Nathan Kutz. Compressed singular value decomposition for image and video processing. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 1880–1888, 2017.

[11] James Bennett, Stan Lanning, et al. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35. New York, 2007.

[12] Austin R. Benson, Rediet Abebe, Michael T. Schaub, Ali Jadbabaie, and Jon Kleinberg. Simplicial closure and higher-order link prediction. *Proceedings of the National Academy of Sciences*, 2018.

[13] Michael W Berry. Large-scale sparse singular value computations. *The International Journal of Supercomputing Applications*, 6(1):13–49, 1992.

[14] Thierry Bertin-Mahieux, Daniel PW Ellis, Brian Whitman, and Paul Lamere. The million song dataset. 2011.

[15] Ranran Bian, Yun Sing Koh, Gillian Dobbie, and Anna Divoli. Network embedding and change modeling in dynamic heterogeneous networks. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 861–864, 2019.

[16] Isaac I Bogoch, Alexander Watts, Andrea Thomas-Bachli, Carmen Huber, Moritz UG Kraemer, and Kamran Khan. Potential for global spread of a novel coronavirus from china. *Journal of travel medicine*, 27(2):taaa011, 2020.

[17] Fedor Borisyuk, Shihai He, Yunbo Ouyang, Morteza Ramezani, Peng Du, Xiaochen Hou, Chengming Jiang, Nitin Pasumarthy, Priya Bannur, Birjodh Tiwana, et al. Lignn: Graph neural networks at linkedin. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 4793–4803, 2024.

[18] Elizabeth Boschee, Jennifer Lautenschlager, Sean O'Brien, Steve Shellman, James Starz, and Michael Ward. ICEWS Coded Event Data, 2015.

[19] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 93–104, 2000.

[20] Rasmus Bro. Parafac. tutorial and applications. *Chemometrics and intelligent laboratory systems*, 38(2):149–171, 1997.

[21] Anna D Broido and Aaron Clauset. Scale-free networks are rare. *Nature communications*, 10(1):1017, 2019.

[22] Tom B Brown. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[23] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.

[24] Benjamin Paul Chamberlain, Sergey Shirobokov, Emanuele Rossi, Fabrizio Frasca, Thomas Markovich, Nils Yannick Hammerla, Michael M Bronstein, and Max Hans-

mire. Graph neural networks for link prediction with subgraph sketching. In *The Eleventh International Conference on Learning Representations*, 2022.

[25] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58, 2009.

[26] Serina Chang, Emma Pierson, Pang Wei Koh, Jaline Gerardin, Beth Redbird, David Grusky, and Jure Leskovec. Mobility network models of covid-19 explain inequities and inform reopening. *Nature*, 589(7840):82–87, 2021.

[27] Sudhanshu Chanpuriya, Ryan A Rossi, Sungchul Kim, Tong Yu, Jane Hoffswell, Nedim Lipka, Shunan Guo, and Cameron Musco. Direct embedding of temporal network edges via time-decayed line graphs. *arXiv preprint arXiv:2210.00032*, 2022.

[28] Jinyin Chen, Xueke Wang, and Xuanheng Xu. Gc-lstm: Graph convolution embedded lstm for dynamic network link prediction. *Applied Intelligence*, pages 1–16, 2022.

[29] Zhengdao Chen, Soledad Villar, Lei Chen, and Joan Bruna. On the equivalence between graph isomorphism testing and function approximation with gnns. *Advances in neural information processing systems*, 32, 2019.

[30] Zhengzhang Chen, William Hendrix, and Nagiza F Samatova. Community-based anomaly detection in evolutionary networks. *Journal of Intelligent Information Systems*, 39(1):59–85, 2012.

[31] Haibin Cheng, Pang-Ning Tan, Christopher Potter, and Steven Klooster. Detection and characterization of anomalies in multivariate time series. In *Proceedings of the 2009 SIAM international conference on data mining*, pages 413–424. SIAM, 2009.

[32] Fan Chung. Laplacians and the cheeger inequality for directed graphs. *Annals of Combinatorics*, 9:1–19, 2005.

[33] Fan RK Chung and Fan Chung Graham. *Spectral graph theory*. Number 92. American Mathematical Soc., 1997.

[34] Andrea Cini, Ivan Marisca, Filippo Maria Bianchi, and Cesare Alippi. Scalable spatiotemporal graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 7218–7226, 2023.

[35] Andrea Cini, Ivan Marisca, Daniele Zambon, and Cesare Alippi. Taming local effects in graph-based spatiotemporal forecasting. *Advances in Neural Information Processing Systems*, 36, 2024.

[36] Our Commons. In the house.

[37] Weilin Cong, Si Zhang, Jian Kang, Baichuan Yuan, Hao Wu, Xin Zhou, Hanghang Tong, and Mehrdad Mahdavi. Do we really need complicated model architectures for temporal networks? In *The Eleventh International Conference on Learning Representations*, 2022.

[38] William Cross. The importance of local party activity in understanding canadian politics: Winning from the ground up in the 2015 federal election: Presidential address to the canadian political science association calgary, 31 may 2016. *Canadian Journal of Political Science/Revue canadienne de science politique*, 49(4):601–620, 2016.

[39] Michał Daniluk and Jacek Dąbrowski. Temporal graph models fail to capture global temporal dynamics. *arXiv preprint arXiv:2309.15730*, 2023.

[40] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[41] Manuel Dileo, Matteo Zignani, and Sabrina Gaito. Durendal: Graph deep learning framework for temporal heterogeneous networks. *arXiv preprint arXiv:2310.00336*, 2023.

[42] Xiaoye Ding, Shenyang Huang, Abby Leung, and Reihaneh Rabbany. Incorporating dynamic flight network in seir to model mobility between populations. *Applied Network Science*, 6(1):1–24, 2021.

[43] Zifeng Ding, Heling Cai, Jingpei Wu, Yunpu Ma, Ruotong Liao, Bo Xiong, and Volker Tresp. zrllm: Zero-shot relational learning on temporal knowledge graphs with large language models. In *NAACL*, 2024.

[44] Zifeng Ding, Yifeng Li, Yuan He, Antonio Norelli, Jingcheng Wu, Volker Tresp, Yunpu Ma, and Michael Bronstein. Dygmamba: Efficiently modeling long-term temporal dependency on continuous-time dynamic graphs with state space models. *arXiv preprint arXiv:2408.04713*, 2024.

[45] Kun Dong, Austin R Benson, and David Bindel. Network density of states. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1152–1161, 2019.

[46] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020.

[47] Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*, 2020.

[48] Vijay Prakash Dwivedi, Ladislav Rampášek, Michael Galkin, Ali Parviz, Guy Wolf, Anh Tuan Luu, and Dominique Beaini. Long range graph benchmark. *Advances in Neural Information Processing Systems*, 35:22326–22340, 2022.

[49] Nathan Eagle, Alex Sandy Pentland, and David Lazer. Inferring friendship network structure by using mobile phone data. *Proceedings of the national academy of sciences*, 106(36):15274–15278, 2009.

[50] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

[51] Paul Erdős and Alfréd Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1):17–60, 1960.

[52] Federico Errica, Marco Podda, Davide Bacciu, and Alessio Micheli. A fair comparison of graph neural networks for graph classification. In *8th International Conference on Learning Representations (ICLR)*, 2020.

[53] Dhivya Eswaran and Christos Faloutsos. Sedanspot: Detecting anomalies in edge streams. In *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2018.

[54] Dhivya Eswaran, Christos Faloutsos, Sudipto Guha, and Nina Mishra. Spotlight: Detecting anomalies in streaming graphs. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, page 1378–1386, New York, NY, USA, 2018. Association for Computing Machinery.

[55] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

[56] James H Fowler. Legislative cosponsorship networks in the us house and senate. *Social Networks*, 28(4):454–465, 2006.

[57] Alan Frieze and Michał Karoński. *Introduction to random graphs*. Cambridge University Press, 2015.

[58] Mostafa Reisi Gahrooei and Kamran Paynabar. Change detection in a dynamic stream of attributed networks. *Journal of Quality Technology*, 50(4):418–430, 2018.

[59] Mikhail Galkin, Xinyu Yuan, Hesham Mostafa, Jian Tang, and Zhaocheng Zhu. Towards foundation models for knowledge graph reasoning. In *The Twelfth International Conference on Learning Representations*, 2024.

[60] Riccardo Gallotti and Marc Barthelemy. The multilayer temporal network of public transport in great britain. *Scientific data*, 2(1):1–8, 2015.

[61] Jianfei Gao and Bruno Ribeiro. On the equivalence between temporal and static equivariant graph representations. In *International Conference on Machine Learning*, pages 7052–7076. PMLR, 2022.

[62] Alberto García-Durán, Sebastijan Dumančić, and Mathias Niepert. Learning sequence encoders for temporal knowledge graph completion. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4816–4821, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[63] Julia Gastinger, Shenyang Huang, Mikhail Galkin, Erfan Loghmani, Ali Parviz, Farimah Poursafaei, Jacob Danovitch, Emanuele Rossi, Ioannis Koutis, Heiner Stuckenschmidt, et al. Tgb 2.0: A benchmark for learning on temporal knowledge graphs and heterogeneous graphs. *arXiv preprint arXiv:2406.09639*, 2024.

[64] Julia Gastinger, Christian Meilicke, Federico Errica, Timo Sztyler, Anett Schuelke, and Heiner Stuckenschmidt. History repeats itself: A baseline for temporal knowledge graph forecasting. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI 2024, Jeju, South Korea*, 2024.

[65] Julia Gastinger, Timo Sztyler, Lokesh Sharma, Anett Schuelke, and Heiner Stuckenschmidt. Comparing apples and oranges? On the evaluation of methods for temporal knowledge graph forecasting. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*, pages 533–549, 2023.

[66] Deborah J Gerner, Philip A Schrodt, Omür Yilmaz, and Rajaa Abu-Jabr. Conflict and mediation event observations (cameo): A new event data framework for the analysis of foreign policy interactions. *International Studies Association, New Orleans*, 2002.

[67] Gene H Golub and Christian Reinsch. Singular value decomposition and least squares solutions. In *Linear Algebra*, pages 134–151. Springer, 1971.

[68] Alessio Gravina and Davide Bacciu. Deep learning for dynamic graphs: Models and benchmarks. *IEEE Transactions on Neural Networks and Learning Systems*, 2024.

[69] Sudipto Guha, Nina Mishra, Gourav Roy, and Okke Schrijvers. Robust random cut forest based anomaly detection on streams. In *International conference on machine learning*, pages 2712–2721. PMLR, 2016.

[70] Jun Guo and Wenwu Zhu. Partial multi-view outlier detection based on collective learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[71] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.

[72] Kenneth M Hall. An r-dimensional quadratic placement algorithm. *Management science*, 17(3):219–229, 1970.

[73] Andrew Halterman, Benjamin E Bagozzi, Andreas Beger, Phil Schrodt, and Grace Scraborough. Plover and polecat: A new political event ontology and dataset. In *International Studies Association Conference Paper*, 2023.

[74] Zhen Han, Peng Chen, Yunpu Ma, and Volker Tresp. Explainable subgraph reasoning for forecasting on temporal knowledge graphs. In *9th International Conference on Learning Representations (ICLR)*, 2021.

[75] Zhen Han, Zifeng Ding, Yunpu Ma, Yujia Gu, and Volker Tresp. Learning neural ordinary equations for forecasting future links on temporal knowledge graphs. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8352–8364, 2021.

[76] Zhen Han, Yunpu Ma, Yuyi Wang, Stephan Günnemann, and Volker Tresp. Graph hawkes neural network for forecasting on temporal knowledge graphs. In Dipanjan Das, Hannaneh Hajishirzi, Andrew McCallum, and Sameer Singh, editors, *Conference on Automated Knowledge Base Construction, AKBC 2020, Virtual, June 22-24, 2020*, 2020.

[77] F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19, 2015.

[78] Richard A Harshman et al. Foundations of the parafac procedure: Models and conditions for an" explanatory" multimodal factor analysis. 1970.

[79] Shigefumi Hata and Hiroya Nakao. Localization of laplacian eigenvectors on random networks. *Scientific reports*, 7(1):1–11, 2017.

[80] Douglas M Hawkins. *Identification of outliers*, volume 11. Springer, 1980.

[81] Balázs Hidasi and Domonkos Tikk. Fast als-based tensor factorization for context-aware recommendation from implicit feedback. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2012, Bristol, UK, September 24-28, 2012. Proceedings, Part II 23*, pages 67–82. Springer, 2012.

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

[83] Paul W Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Social networks*, 5(2):109–137, 1983.

[84] Petter Holme. Epidemiologically optimal static networks from temporal network data. *PLoS computational biology*, 9(7):e1003142, 2013.

[85] Petter Holme. Modern temporal network theory: a colloquium. *The European Physical Journal B*, 88:1–30, 2015.

[86] Yupeng Hou, Jiacheng Li, Zhankui He, An Yan, Xiusi Chen, and Julian J McAuley. Bridging language and items for retrieval and recommendation. *CoRR*, 2024.

[87] Weihua Hu, Matthias Fey, Hongyu Ren, Maho Nakata, Yuxiao Dong, and Jure Leskovec. Ogb-lsc: A large-scale challenge for machine learning on graphs. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[88] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.

[89] Chao Huang, Chuxu Zhang, Jiashu Zhao, Xian Wu, Dawei Yin, and Nitesh Chawla. Mist: A multiview and multimodal spatial-temporal learning framework for city-wide abnormal event forecasting. In *The World Wide Web Conference*, pages 717–728, 2019.

[90] Leo Huang, Andrew J Graven, and David Bindel. Density of states graph kernels. In *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*, 2021.

[91] Shenyang Huang, Samy Coulombe, Yasmeen Hitti, Reihaneh Rabbany, and Guillaume Rabusseau. Laplacian change point detection for single and multi-view dynamic graphs. *ACM Transactions on Knowledge Discovery from Data*, 18(3):1–32, 2024.

[92] Shenyang Huang, Jacob Danovitch, Guillaume Rabusseau, and Reihaneh Rabbany. Fast and attributed change detection on dynamic graphs with density of states. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 15–26. Springer, 2023.

[93] Shenyang Huang, Yasmeen Hitti, Guillaume Rabusseau, and Reihaneh Rabbany. Laplacian change point detection for dynamic graphs. In *Proceedings of the 26th*

ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pages 349–358, 2020.

[94] Shenyang Huang, Farimah Poursafaei, Jacob Danovitch, Matthias Fey, Weihua Hu, Emanuele Rossi, Jure Leskovec, Michael Bronstein, Guillaume Rabusseau, and Reihaneh Rabbany. Temporal graph benchmark for machine learning on temporal graphs. In *37th Conference on Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2023.

[95] Shenyang Huang, Farimah Poursafaei, Reihaneh Rabbany, Guillaume Rabusseau, and Emanuele Rossi. Utg: Towards a unified view of snapshot and event based models for temporal graphs. *arXiv preprint arXiv:2407.12269*, 2024.

[96] Xuanwen Huang, Yang Yang, Yang Wang, Chunping Wang, Zhisheng Zhang, Jiarong Xu, and Lei Chen. DGraph: A large-scale financial dataset for graph anomaly detection. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

[97] Tsuyoshi Idé and Hisashi Kashima. Eigenspace-based anomaly detection in computer systems. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 440–449. ACM, 2004.

[98] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002.

[99] Tengfei Ji, Dongqing Yang, and Jun Gao. Incremental local evolutionary outlier detection for dynamic social networks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 1–15. Springer, 2013.

[100] Yugang Ji, Tianrui Jia, Yuan Fang, and Chuan Shi. Dynamic heterogeneous graph embedding via heterogeneous hawkes process. In *Machine Learning and Knowledge Discovery in Databases. Research Track: European Conference, ECML PKDD 2021, Bilbao, Spain, September 13–17, 2021, Proceedings, Part I 21*, pages 388–403. Springer, 2021.

[101] Lekang Jiang, Caiqi Zhang, Farimah Poursafaei, and Shenyang Huang. Towards temporal edge regression: A case study on agriculture trade between nations. *arXiv preprint arXiv:2308.07883*, 2023.

[102] Ming Jin, Yuan-Fang Li, and Shirui Pan. Neural temporal walks: Motif-aware representation learning on continuous-time dynamic graphs. In *Advances in Neural Information Processing Systems*, 2022.

[103] Woojeong Jin, Meng Qu, Xisen Jin, and Xiang Ren. Recurrent event network: Autoregressive structure inferenceover temporal knowledge graphs. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6669–6683, 2020.

[104] Seyed Mehran Kazemi, Rishab Goel, Kshitij Jain, Ivan Kobyzev, Akshay Sethi, Peter Forsyth, and Pascal Poupart. Representation learning for dynamic graphs: A survey. *The Journal of Machine Learning Research*, 21(1):2648–2720, 2020.

[105] Alexandra Kerr. A historical timeline of covid-19 in new york city, 2020.

[106] Byung-Hoon Kim, Jungwon Choi, EungGu Yun, Kyungsang Kim, Xiang Li, and Juho Lee. Large-scale graph representation learning of dynamic brain connectome with transformers. *arXiv preprint arXiv:2312.14939*, 2023.

[107] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

[108] Rage Uday Kiran, Abinash Maharana, and Krishna Reddy Polepalli. A novel explainable link forecasting framework for temporal knowledge graphs using time-relaxed cyclic and acyclic rules. In *Proceedings of the 27th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), Part I*, pages 264–275, 2023.

[109] Niko Kiukkonen, Jan Blom, Olivier Dousse, Daniel Gatica-Perez, and Juha Laurila. Towards rich mobile phone datasets: Lausanne data collection campaign. *Proc. ICPS, Berlin*, 68, 2010.

[110] Kerstin Klaser, Blazej Banaszewski, Samuel Maddrell-Mander, Callum McLean, Luis Müller, Ali Parviz, Shenyang Huang, and Andrew W Fitzgibbon. Minimol: A parameter-efficient foundation model for molecular learning. In *ICML 2024 Workshop on Efficient and Accessible Foundation Models for Biological Discovery*, 2024.

[111] Danai Koutra, Evangelos E Papalexakis, and Christos Faloutsos. Tensorsplat: Spotting latent anomalies in time. In *2012 16th Panhellenic Conference on Informatics*, pages 144–149. IEEE, 2012.

[112] Danai Koutra, Neil Shah, Joshua T Vogelstein, Brian Gallagher, and Christos Faloutsos. Deltacon: Principled massive-graph similarity function with attribution. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 10(3):1–43, 2016.

[113] Danai Koutra, Joshua T Vogelstein, and Christos Faloutsos. Deltacon: A principled massive-graph similarity function. In *Proceedings of the 2013 SIAM International Conference on Data Mining*, pages 162–170. SIAM, 2013.

[114] Devin Kreuzer, Dominique Beaini, Will Hamilton, Vincent Létourneau, and Prudencio Tossou. Rethinking graph transformers with spectral attention. *Advances in Neural Information Processing Systems*, 34:21618–21629, 2021.

[115] Srijan Kumar, Xikun Zhang, and Jure Leskovec. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1269–1278, 2019.

[116] Julien Leblay and Melisachew Wudage Chekol. Deriving validity time in knowledge graph. In Pierre-Antoine Champin, Fabien Gandon, Mounia Lalmas, and Panagiotis G. Ipeirotis, editors, *Companion of the The Web Conference 2018 on The*

Web Conference 2018, WWW 2018, Lyon, France, April 23-27, 2018, pages 1771–1776. ACM, 2018.

[117] Kalev Leetaru and Philip A Schrodt. Gdelt: Global data on events, location, and tone, 1979–2012. In *ISA annual convention*, pages 1–49. Citeseer, 2013.

[118] YANG Leshanshui, Clément Chatelain, and Sébastien Adam. Dspgnn: Bringing spectral design to discrete time dynamic graph neural networks for edge regression. In *Temporal Graph Learning Workshop@ NeurIPS 2023*.

[119] Abby Leung, Xiaoye Ding, Shenyang Huang, and Reihaneh Rabbany. Contact graph epidemic modelling of covid-19 for transmission and intervention strategies. *arXiv preprint arXiv:2010.03081*, 2020.

[120] Ce Li, Rongpei Hong, Xovee Xu, Goce Trajcevski, and Fan Zhou. Simplifying temporal heterogeneous network for continuous-time link prediction. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 1288–1297, 2023.

[121] Dongyuan Li, Shiyin Tan, Ying Zhang, Ming Jin, Shirui Pan, Manabu Okumura, and Renhe Jiang. Dyg-mamba: Continuous state space modeling on dynamic graphs. *arXiv preprint arXiv:2408.06966*, 2024.

[122] Zixuan Li, Saiping Guan, Xiaolong Jin, Weihua Peng, Yajuan Lyu, Yong Zhu, Long Bai, Wei Li, Jiafeng Guo, and Xueqi Cheng. Complex evolutional pattern learning for temporal knowledge graph reasoning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL), Volume 2: Short Papers*, pages 290–296, 2022.

[123] Zixuan Li, Xiaolong Jin, Saiping Guan, Wei Li, Jiafeng Guo, Yuanzhuo Wang, and Xueqi Cheng. Search from history and reason for future: Two-stage reasoning on

temporal knowledge graphs. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL/IJCNLP), Volume 1: Long Papers*, pages 4732–4743, 2021.

[124] Zixuan Li, Xiaolong Jin, Wei Li, Saiping Guan, Jiafeng Guo, Huawei Shen, Yuanzhuo Wang, and Xueqi Cheng. Temporal knowledge graph reasoning based on evolutional representation learning. In *The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2021.

[125] Qika Lin, Jun Liu, Rui Mao, Fangzhi Xu, and Erik Cambria. TECHS: Temporal logical graph networks for explainable extrapolation reasoning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL), Volume 1: Long Papers*, pages 1281–1293, 2023.

[126] Kangzheng Liu, Feng Zhao, Guandong Xu, Xianzhi Wang, and Hai Jin. RETIA: Relation-entity twin-interact aggregation for temporal knowledge graph extrapolation. In *39th IEEE International Conference on Data Engineering (ICDE)*, pages 1761–1774, 2023.

[127] Shuwen Liu, Bernardo Grau, Ian Horrocks, and Egor Kostylev. Indigo: Gnn-based inductive knowledge graph completion using pair-wise encoding. *Advances in Neural Information Processing Systems*, 34:2034–2045, 2021.

[128] Yushan Liu, Yunpu Ma, Marcel Hildebrandt, Mitchell Joblin, and Volker Tresp. TLogic: Temporal logical rules for explainable link forecasting on temporal knowledge graphs. In *36th Conference on Artificial Intelligence (AAAI)*, pages 4120–4127, 2022.

[129] Yuxuan Liu, Yijun Mo, Zhengyu Chen, and Huiyu Liu. LogE-Net: Logic evolution network for temporal knowledge graph forecasting. In *International Conference on Artificial Neural Networks (ICANN)*, pages 472–485, 2023.

[130] Erfan Loghmani and MohammadAmin Fazli. Effect of choosing loss function when using t-batching for representation learning on dynamic networks. *CoRR*, abs/2308.06862, 2023.

[131] Antonio Longa, Veronica Lachi, Gabriele Santin, Monica Bianchini, Bruno Lepri, Pietro Lio, Andrea Passerini, et al. Graph neural networks for temporal graphs: State of the art, open challenges, and opportunities. *Transactions on Machine Learning Research*, 2023.

[132] Rui Luo and Vikram Krishnamurthy. Fréchet-statistics-based change point detection in dynamic social networks. *IEEE Transactions on Computational Social Systems*, 2023.

[133] Yuhong Luo and Pan Li. Neighborhood-aware scalable temporal network representation learning. In *Learning on Graphs Conference*, pages 1–1. PMLR, 2022.

[134] Yuhong Luo and Pan Li. No need to look back: An efficient and scalable approach for temporal network representation learning. *arXiv preprint arXiv:2402.01964*, 2024.

[135] Farzaneh Mahdisoltani, Joanna Asia Biega, and Fabian M. Suchanek. Yago3: A knowledge base from multilingual wikipedias. In *CIDR*, 2015.

[136] Alex Marland. What is a political brand?: Justin trudeau and the theory of political branding. In *annual meeting of the Canadian Communication Association and the Canadian Political Science Association, University of Victoria, British Columbia, June*, volume 6, 2013.

[137] Xin Mei, Libin Yang, Xiaoyan Cai, and Zuowei Jiang. An adaptive logical rule embedding model for inductive reasoning over temporal knowledge graphs. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7304–7316, 2022.

[138] Pedro Mercado, Antoine Gautier, Francesco Tudisco, and Matthias Hein. The power mean laplacian for multilayer graph clustering. In *International Conference on Artificial Intelligence and Statistics*, pages 1828–1838, 2018.

[139] Alessio Micheli. Neural network for graphs: A contextual constructive approach. *IEEE Transactions on Neural Networks*, 20(3):498–511, 2009.

[140] Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. Network motifs: simple building blocks of complex networks. *Science*, 298:824–827, 2002.

[141] Amirhossein Nadiri and Frank W Takes. A large-scale temporal analysis of user lifespan durability on the reddit social media platform. In *Companion Proceedings of the Web Conference 2022*, pages 677–685, 2022.

[142] Masataka Nagao, Eriko Segawa, and Yusuke Sakumoto. The effect of small eigenvalues on the effectivity of laplacian anomaly detection of dynamic networks. In *International Conference on Intelligent Networking and Collaborative Systems*, pages 200–210. Springer, 2023.

[143] Mark EJ Newman and Aaron Clauset. Structure and inference in annotated networks. *Nature communications*, 7(1):1–11, 2016.

[144] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33, 2015.

[145] Xavier Olive. Traffic, a toolbox for processing and analysing air traffic data. *Journal of Open Source Software*, 4(39):1518–1, 2019.

[146] Jukka-Pekka Onnela, Daniel J Fenn, Stephen Reid, Mason A Porter, Peter J Mucha, Mark D Fricker, and Nick S Jones. Taxonomies of networks from community structure. *Physical Review E*, 86(3):036104, 2012.

[147] Muberra Ozmen, Florence Regol, and Thomas Markovich. Benchmarking edge regression on temporal networks. *Journal of Data-centric Machine Learning Research*, 2024.

[148] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[149] Trishan Panch, Peter Szolovits, and Rifat Atun. Artificial intelligence, machine learning and health systems. *Journal of global health*, 8(2), 2018.

[150] Pietro Panzarasa, Tore Opsahl, and Kathleen M Carley. Patterns and dynamics of users' behavior and interaction: Network analysis of an online community. *Journal of the American Society for Information Science and Technology*, 60(5):911–932, 2009.

[151] Evangelos E Papalexakis, Leman Akoglu, and Dino Ience. Do more views of a graph help? community detection and clustering in multi-graphs. In *Proceedings of the 16th International Conference on Information Fusion*, pages 899–905. IEEE, 2013.

[152] Ashwin Paranjape, Austin R Benson, and Jure Leskovec. Motifs in temporal networks. In *Proceedings of the tenth ACM international conference on web search and data mining*, pages 601–610, 2017.

[153] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao Schardl, and Charles Leiserson. Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 5363–5370, 2020.

[154] R Pascanu. On the difficulty of training recurrent neural networks. *arXiv preprint arXiv:1211.5063*, 2013.

[155] Leto Peel and Aaron Clauset. Detecting change points in the large-scale structure of evolving networks. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[156] Edwin JG Pitman. Significance tests which may be applied to samples from any populations. *Supplement to the Journal of the Royal Statistical Society*, 4(1):119–130, 1937.

[157] Tim Poštuvan, Claas Grohnfeldt, Michele Russo, and Giulio Lovisotto. Learning-based link anomaly detection in continuous-time dynamic graphs. *arXiv preprint arXiv:2405.18050*, 2024.

[158] Farimah Poursafaei, Shenyang Huang, Kellin Pelrine, and Reihaneh Rabbany. Towards better evaluation for dynamic link prediction. *Advances in Neural Information Processing Systems*, 35:32928–32941, 2022.

[159] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[160] Ladislav Rampášek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. *Advances in Neural Information Processing Systems*, 35:14501–14515, 2022.

[161] Stephen Ranshous, Shitian Shen, Danai Koutra, Steve Harenberg, Christos Faloutsos, and Nagiza F Samatova. Anomaly detection in dynamic networks: a survey. *Wiley Interdisciplinary Reviews: Computational Statistics*, 7(3):223–247, 2015.

[162] Jérémie Rappaz, Julian McAuley, and Karl Aberer. Recommendation on live-streaming platforms: Dynamic availability and repeat consumption. In *Proceedings of the 15th ACM Conference on Recommender Systems*, pages 390–399, 2021.

[163] Jakub Reha, Giulio Lovisotto, Michele Russo, Alessio Gravina, and Claas Grohnfeldt. Anomaly detection in continuous-time temporal provenance graphs. In *Temporal Graph Learning Workshop@ NeurIPS 2023*, 2023.

[164] Andrea Rossi, Denilson Barbosa, Donatella Firmani, Antonio Matinata, and Paolo Merialdo. Knowledge graph embedding for link prediction: A comparative analysis. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 15(2):1–49, 2021.

[165] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637*, 2020.

[166] Benedek Rozemberczki, Paul Scherer, Yixuan He, George Panagopoulos, Alexander Riedel, Maria Astefanoaei, Oliver Kiss, Ferenc Beres, Guzman Lopez, Nicolas Collignon, et al. Pytorch geometric temporal: Spatiotemporal signal processing with neural machine learning models. In *Proceedings of the 30th ACM international conference on information & knowledge management*, pages 4564–4573, 2021.

[167] Benedek Rozemberczki, Paul Scherer, Yixuan He, George Panagopoulos, Alexander Riedel, Maria Astefanoaei, Oliver Kiss, Ferenc Beres, Guzman Lopez, Nicolas Collignon, and Rik Sarkar. PyTorch Geometric Temporal: Spatiotemporal Signal Processing with Neural Machine Learning Models. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management*, page 4564–4573, 2021.

[168] Awwal Mohammed Rufai, Gholamreza Anbarjafari, and Hasan Demirel. Lossy image compression using singular value decomposition and wavelet difference reduction. *Digital signal processing*, 24:117–123, 2014.

[169] S Salcedo-Sanz, D Casillas-Pérez, J Del Ser, C Casanova-Mateo, L Cuadra, M Piles, and G Camps-Valls. Persistence in complex systems. *Physics Reports*, 957:1–73, 2022.

[170] Saurabh Sawlani, Lingxiao Zhao, and Leman Akoglu. Fast attributed graph embedding via density of states. In *2021 IEEE International Conference on Data Mining (ICDM)*, 2021.

[171] Grace I. Scarborough, Benjamin E. Bagozzi, Andreas Beger, John Berrie, Andrew Halterman, Philip A. Schrodt, and Jevon Spivey. POLECAT Weekly Data, 2023.

[172] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

[173] Matthias Schäfer, Martin Strohmeier, Vincent Lenders, Ivan Martinovic, and Matthias Wilhelm. Bringing up opensky: A large-scale ads-b sensor network for research. In *IPSN-14 Proceedings of the 13th International Symposium on Information Processing in Sensor Networks*, pages 83–94. IEEE, 2014.

[174] Ingo Scholtes. When is a network a network? multi-order graphical model selection in pathways and temporal networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1037–1046, 2017.

[175] Kiarash Shamsi, Yulia R. Gel, Murat Kantarcioglu, and Cuneyt G. Akcora. Chartalist: Labeled graph datasets for utxo and account-based blockchains. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, November 29-December 1, 2022, New Orleans, LA, USA*, pages 1–14, 2022.

[176] Kiarash Shamsi, Farimah Poursafaei, Shenyang Huang, Bao Tran Gia Ngo, Baris Coskunuzer, and Cuneyt Gurcan Akcora. Graphpulse: Topological representations for temporal graph property prediction. In *The Twelfth International Conference on Learning Representations*, 2024.

[177] Kiarash Shamsi, Friedhelm Victor, Murat Kantarcioglu, Yulia Gel, and Cuneyt G Akcora. Chartalist: Labeled graph datasets for utxo and account-based blockchains. *Advances in Neural Information Processing Systems*, 35:34926–34939, 2022.

[178] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *Workshop on Relational*

Representation Learning, 32nd Conference on Neural Information Processing Systems (NeurIPS), 2018.

[179] Xiang-Rong Sheng, De-Chuan Zhan, Su Lu, and Yuan Jiang. Multi-view anomaly detection: Neighborhood in locality matters. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4894–4901, 2019.

[180] Andrew Shilliday, Jennifer Lautenschlager, et al. Data for a worldwide icews and ongoing research. *Advances in Design for Cross-Cultural Activities*, 455, 2012.

[181] Razieh Shirzadkhani, Shenyang Huang, Elahe Kooshafar, Reihaneh Rabbany, and Farimah Poursafaei. Temporal graph analysis with tgx. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pages 1086–1089, 2024.

[182] Razieh Shirzadkhani, Shenyang Huang, Abby Leung, and Reihaneh Rabbany. Static graph approximations of dynamic contact networks for epidemic forecasting. *Scientific Reports*, 14(1):11696, 2024.

[183] Razieh Shirzadkhani, Tran Gia Bao Ngo, Kiarash Shamsi, Shenyang Huang, Farimah Poursafaei, Poupak Azad, Reihaneh Rabbany, Baris Coskunuzer, Guillaume Rabusseau, and Cuneyt Gurcan Akcora. Towards neural scaling laws for foundation models on temporal graphs. *arXiv preprint arXiv:2406.10426*, 2024.

[184] Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-June (Paul) Hsu, and Kuansan Wang. An overview of microsoft academic service (MAS) and applications. In *the 24th International Conference on World Wide Web*. ACM Press, 2015.

[185] Joakim Skarding, Matthew Hellmich, Bogdan Gabrys, and Katarzyna Musial. A robust comparative analysis of graph neural networks on dynamic link prediction. *IEEE Access*, 10:64146–64160, 2022.

[186] Yunkyu Sohn, Jonghee Park, et al. Bayesian approach to multilayer stochastic blockmodel and network changepoint detection. *Netw. Sci.*, 5(2):164–186, 2017.

[187] Amauri Souza, Diego Mesquita, Samuel Kaski, and Vikas Garg. Provably expressive temporal graph networks. *Advances in Neural Information Processing Systems*, 35:32257–32269, 2022.

[188] Daniel A Spielman. Spectral graph theory and its applications. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*. IEEE, 2007.

[189] Haohai Sun, Jialun Zhong, Yunpu Ma, Zhen Han, and Kun He. Timetraveler: Reinforcement learning for temporal knowledge graph forecasting. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8306–8319, 2021.

[190] Zhiqing Sun, Shikhar Vashishth, Soumya Sanyal, Partha P. Talukdar, and Yiming Yang. A re-evaluation of knowledge graph completion methods. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 5516–5522, 2020.

[191] Toyotaro Suzumura, Hiroki Kanezashi, Mishal Dholakia, Euma Ishii, Sergio Alvarez Napagao, Raquel Pérez-Arnal, and Dario Garcia-Gasulla. The impact of covid-19 on flight networks. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 2443–2452. IEEE, 2020.

[192] Lei Tang, Xufei Wang, and Huan Liu. Community detection in multi-dimensional networks. Technical report, Arizona State University, Department of Computer Science and Engineering, 2010.

[193] Wei Tang, Zhengdong Lu, and Inderjit S Dhillon. Clustering with multiple graphs. In *2009 Ninth IEEE International Conference on Data Mining*, pages 1016–1021. IEEE, 2009.

[194] Xian Teng, Yu-Ru Lin, and Xidao Wen. Anomaly detection in dynamic networks using multi-view time-series hypersphere learning. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 827–836, 2017.

[195] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. Dyrep: Learning representations over dynamic graphs. In *International conference on learning representations*, 2019.

[196] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

[197] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[198] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake Vand erPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1. 0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 2020.

[199] Renato Vizuete, Federica Garin, and Paolo Frasca. The laplacian spectrum of large graphs sampled from graphons. *arXiv preprint arXiv:2004.09177*, 2020.

[200] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[201] Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.

[202] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Glue: A multi-task benchmark and analysis platform for natural lan-

guage understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, 2018.

[203] Baoxin Wang, Qingye Meng, Ziyue Wang, Honghong Zhao, Dayong Wu, Wanxiang Che, Shijin Wang, Zhigang Chen, and Cong Liu. Interht: Knowledge graph embeddings by interaction between head and tail entities. *arXiv preprint arXiv:2202.04897*, 2022.

[204] Junhao Wang, Sacha Levy, Ren Wang, Aayushi Kulshrestha, and Reihaneh Rabbany. Scg: Spotting coordinated groups in social media. *arXiv preprint arXiv:1910.07130*, 2019.

[205] Lu Wang, Xiaofu Chang, Shuang Li, Yunfei Chu, Hui Li, Wei Zhang, Xiaofeng He, Le Song, Jingren Zhou, and Hongxia Yang. Tcl: Transformer-based dynamic graph modelling via contrastive learning. *arXiv preprint arXiv:2105.07944*, 2021.

[206] Yanbang Wang, Yen-Yu Chang, Yunyu Liu, Jure Leskovec, and Pan Li. Inductive representation learning in temporal networks via causal anonymous walks. In *International Conference on Learning Representations*, 2020.

[207] Yu Wang, Aniket Chakrabarti, David Sivakoff, and Srinivasan Parthasarathy. Fast change point detection on dynamic social networks. *arXiv preprint arXiv:1705.07325*, 2017.

[208] Boris Weisfeiler and Andrei Leman. The reduction of a graph to canonical form and the algebra which appears therein. *NTI, Series*, 2(9):12–16, 1968.

[209] Qingsong Wen, Tian Zhou, Chaoli Zhang, Weiqi Chen, Ziqing Ma, Junchi Yan, and Liang Sun. Transformers in time series: a survey. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, pages 6778–6786, 2023.

[210] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[211] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. Graph neural networks in recommender systems: a survey. *ACM Computing Surveys*, 55(5):1–37, 2022.

[212] Lianghao Xia and Chao Huang. Anygraph: Graph foundation model in the wild. 2024.

[213] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. Inductive representation learning on temporal graphs. 2020.

[214] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2018.

[215] Yi Xu, Junjie Ou, Hui Xu, and Luoyi Fu. Temporal knowledge graph reasoning with historical contrastive learning. In *37th Conference on Artificial Intelligence (AAAI)*, pages 4765–4773, 2023.

[216] Hansheng Xue, Luwei Yang, Wen Jiang, Yi Wei, Yi Hu, and Yu Lin. Modeling dynamic heterogeneous network for link prediction using hierarchical attention with temporal rnn. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2020, Ghent, Belgium, September 14–18, 2020, Proceedings, Part I*, pages 282–298. Springer, 2021.

[217] Menglin Yang, Min Zhou, Marcus Kalander, Zengfeng Huang, and Irwin King. Discrete-time temporal network embedding via implicit hierarchical learning in hyperbolic space. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1975–1985, 2021.

[218] Menglin Yang, Min Zhou, Marcus Kalander, Zengfeng Huang, and Irwin King. Discrete-time temporal network embedding via implicit hierarchical learning in hyperbolic space. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1975–1985, 2021.

[219] Haoteng Yin, Muhan Zhang, Yanbang Wang, Jianguo Wang, and Pan Li. Algorithm and system co-design for efficient subgraph-based graph representation learning. *Proceedings of the VLDB Endowment*, 15(11):2788–2796, 2022.

[220] Ying Yin, Li-Xin Ji, Jian-Peng Zhang, and Yu-Long Pei. Dhne: Network representation learning method for dynamic heterogeneous networks. *IEEE Access*, 7:134782–134792, 2019.

[221] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? *Advances in neural information processing systems*, 34:28877–28888, 2021.

[222] Minji Yoon, Bryan Hooi, Kijung Shin, and Christos Faloutsos. Fast and accurate anomaly detection in dynamic graphs with a two-pronged approach. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 647–657, 2019.

[223] Jiaxuan You, Tianyu Du, and Jure Leskovec. Roland: graph learning framework for dynamic graphs. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2358–2366, 2022.

[224] Le Yu, Leilei Sun, Bowen Du, and Weifeng Lv. Towards better dynamic graph learning: New architecture and unified library. *arXiv preprint arXiv:2303.13047*, 2023.

[225] Rose Yu, Huida Qiu, Zhen Wen, ChingYung Lin, and Yan Liu. A survey on social media anomaly detection. *ACM SIGKDD Explorations Newsletter*, 18(1):1–14, 2016.

[226] Chongjian Yue, Lun Du, Qiang Fu, Wendong Bi, Hengyu Liu, Yu Gu, and Di Yao. Htgn-btw: Heterogeneous temporal graph network with bi-time-window training strategy for temporal link prediction. *arXiv preprint arXiv:2202.12713*, 2022.

[227] Sina Zamani, Tejaswi Nanjundaswamy, and Kenneth Rose. Frequency domain singular value decomposition for efficient spatial audio coding. In *2017 IEEE Workshop*

on Applications of Signal Processing to Audio and Acoustics (WASPAA), pages 126–130. IEEE, 2017.

[228] Mengqi Zhang, Yuwei Xia, Qiang Liu, Shu Wu, and Liang Wang. Learning latent relations for temporal knowledge graph reasoning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL), Volume 1: Long Papers*, pages 12617–12631, 2023.

[229] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. *Advances in neural information processing systems*, 31, 2018.

[230] Muhan Zhang, Pan Li, Yinglong Xia, Kai Wang, and Long Jin. Labeling trick: A theory of using graph neural networks for multi-node representation learning, 2020.

[231] Xiao-Dong Zhang. The laplacian eigenvalues of graphs: a survey. *arXiv preprint arXiv:1111.2897*, 2011.

[232] Xiaohui Zhang, Yanbo Wang, Xiyuan Wang, and Muhan Zhang. Efficient neural common neighbor for temporal graph link prediction. *arXiv preprint arXiv:2406.07926*, 2024.

[233] Jianan Zhao, Hesham Mostafa, Mikhail Galkin, Michael Bronstein, Zhaocheng Zhu, and Jian Tang. Graphany: A foundation model for node classification on any graph. *ArXiv*, abs/2405.20445, 2024.

[234] Hongkuan Zhou, Da Zheng, Israt Nisa, Vasileios Ioannidis, Xiang Song, and George Karypis. Tgl: A general framework for temporal gnn training on billion-scale graphs. *arXiv preprint arXiv:2203.14883*, 2022.

[235] Cunchao Zhu, Muhao Chen, Changjun Fan, Guangquan Cheng, and Yan Zhang. Learning from history: Modeling temporal knowledge graphs with sequential copy-generation networks. In *35th Conference on Artificial Intelligence (AAAI)*, pages 4732–4740, 2021.

[236] Yifan Zhu, Fangpeng Cong, Dan Zhang, Wenwen Gong, Qika Lin, Wenzheng Feng, Yuxiao Dong, and Jie Tang. Wingnn: Dynamic graph neural networks with random gradient aggregation window. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3650–3662, 2023.

[237] Eric R Ziegel. Standard probability and statistics tables and formulae. *Technometrics*, 43(2):249, 2001.