université
PARIS-SACLAY

Habilitation à diriger des recherches

A Machine Learning Tour in Network Science

Habilitation à diriger des recherches
de l'Université Paris-Saclay

Spécialité: **Informatique mathématique**

présentée et soutenue à Gif-sur-Yvette, le 12 avril 2024, par

Fragkiskos MALLIAROS

Composition du jury

Paul-Henry COURNÈDE Professeur, CentraleSupélec, Université Paris-Saclay	Président
Francesco BONCHI Directeur de recherche, Centai Institute	Rapporteur
Srinivasan PARTHASARATHY Professeur, Ohio State University	Rapporteur
Céline ROBARDET Professeure, INSA Lyon, Université de Lyon	Rapportrice
Florence D'ALCHÉ-BUC Professeure, Télécom Paris, Institut Polytechnique de Paris	Examinatrice
Santo FORTUNATO Professeur, Indiana University	Examinateur

RÉSUMÉ

Les graphes, également appelés réseaux, sont des structures de données largement utilisées pour modéliser des systèmes complexes dans divers domaines, des sciences sociales à la biologie et à l'ingénierie. Leur force réside dans leur capacité à représenter les relations entre entités, telles que les amitiés dans les réseaux sociaux ou les interactions protéines dans les réseaux biologiques. En plus de leurs capacités de modélisation, les graphes offrent un cadre mathématique qui sert à analyser, comprendre et faire des prédictions à partir d'ensembles de données du monde réel. Ce manuscrit HDR présente une partie de mes contributions de recherche dans le domaine de l'apprentissage, des représentations des graphes et de ses applications à la science des réseaux. Il présente les travaux menés après avoir rejoint CentraleSupélec, Université Paris-Saclay en 2017. La première partie du manuscrit analyse les techniques de plonger les nœuds en préservant la structure qui utilisent les marches aléatoires. La deuxième partie aborde le défi du développement des modèles d'apprentissage de représentation pour les graphes multicouches et hétérogènes, en soulignant les applications issues du domaine de la biologie computationnelle. La troisième partie se focalise sur la conception de modèles des reseaux de neurones en graphes expressifs et explicables. Finalement, la dernière partie étudie l'application de l'apprentissage de la représentation de graphe afin d'aborder les problèmes de l'apprentissage et de la maximisation de l'influence sociale dans des réseaux complexes.

ABSTRACT

Graphs, also known as networks, are widely used data structures for modeling complex systems in various fields, from the social sciences to biology and engineering. The strength lies in their ability to represent relationships between entities, such as friendships in social networks or protein interactions in biological networks. In addition to their modeling capabilities, graphs offer a mathematical framework to analyze, understand, and make predictions from real-world datasets. This HDR manuscript presents part of my research contributions to the field of graph representation learning and its applications in network science, focusing on the work conducted after joining CentraleSupélec, Université Paris-Saclay in 2017. The first part of the manuscript explores structure-preserving node embedding techniques that leverage random walks. The second part addresses the challenge of developing graph representation learning models for multilayer and heterogeneous graphs, with a specific focus on applications arising from the domain of computational biology. The third part delves into the design of expressive and explainable graph neural network models. Finally, the last part investigates the application of graph representation learning to tackle the well-studied problems of social influence learning and maximization in complex networks.

To my late grandparents Fragkiskos and Nikos
for the life lessons they taught me.

CONTENTS

1

INTRODUCTION

Graphs, or networks, are omnipresent, as data from various disciplines can naturally be represented using graph structures. Characteristic examples include social, information, or biological networks [New18; MFD20]. Furthermore, graphs not only serve as effective models for data representation but have proven valuable in machine learning tasks. For instance, there could be an interest in predicting the missing structure or the role of a group of genes in gene regulatory networks. Likewise, tasks may involve identifying patients with similar phenotypes and symptoms within a patient similarity network or uncovering influential users in social networks, with applications ranging from misinformation detection to viral marketing. Therefore, developing machine learning algorithms that integrate graph structure information into the learning model is a pivotal task offering numerous interdisciplinary applications.

Consider, for instance, the case of friendship recommendations in social networks (also known as the link prediction problem). Here, we need to obtain a meaningful representation of individuals and their proximity to determine whether two unlinked users are similar. However, their proximity is not fully captured by graph statistics (e.g., centrality criteria) or, more generally, other handcrafted features extracted from the graph. Graph representation learning, a prominent recent paradigm, aims at finding vector representations, also known as embeddings, at the node, subgraph, or graph level. It allows for preserving the essential structure of the network and its various properties in the lower dimensional space representations. The problem is typically expressed as a learning task (unsupervised, semi-supervised, or self-supervised), where proximity relationships in the learned space reflect the structure and properties of the original graph [Ham20].

The graph representation learning framework gives rise to a class of flexible algorithms that benefit from the advances in network science, graph mining and relational learning, deep learning, signal processing, and optimization. These algorithms are lately widespread in numerous fields and application domains, including recommender systems, computational chemistry, bioinformatics, natural language processing, and computer vision. Graph representation learning approaches are founded on a solid basis to address challenging prediction tasks, combining the strengths of deep representation learning with the relational inductive bias brought by graphs.

Developing algorithms that learn informative graph representations poses various challenges. First, the models should capture the inherent complex structure of the data involved. Real-world networks, such as those in biology and online social media platforms, can be characterized by structure and dynamics at multiple resolutions. At lower levels, nodes interact with each other, creating connections (i.e., edges) and sharing common node attributes; meanwhile, at the mesoscale, nodes form modules that capture latent communities; and, at a higher level, underlying processes control the way that networks shape and evolve forming intricate node relationships with varying degrees of importance and influence on the graph. Second, in our era of data deluge, we need to handle massive-scale graphs consisting of millions of nodes and edges. Third, the learning process often relies on limited training data, while it should further ensure model generalization capabilities. Fourth, the models should allow for generating human-

interpretable predictions, increasing practitioners' trust. Finally, a remaining challenge is leveraging the representations to deal with real-life applications in emerging domains.

1.1 OVERVIEW OF CONTRIBUTIONS

The work presented in this HDR manuscript focuses on a selection of my research contributions in graph representation learning to address the aforementioned challenges. The research deals with the design of expressive and scalable models and their practical application in interdisciplinary network science tasks, namely identifying influential spreaders in complex networks and analyzing network data in bioinformatics. In the rest of this introduction, I present an overview of the research contributions, summarized schematically in Fig. 1.1.

1.1.1 *Previous Research*

Before presenting the works covered in this manuscript, I will briefly overview my previous research contributions during my Ph.D. and postdoctoral periods (2012–2017). My doctoral thesis centered around the concept of core decomposition on graphs, a well-studied topic in graph mining and network science [Mal15]. Informally, the k-core decomposition is a threshold-based hierarchical decomposition of a graph into nested subgraphs. We explored methodologies that utilized the core decomposition and its variants to study the engagement dynamics of social graphs [MV13b; MV15], detect influential spreaders [MRV16], and categorize documents in a graph-based approach to text analytics [MS15; SMV18]. Additional lines of work in this direction conducted during and after my Ph.D. include frameworks for core-driven graph clustering [Gia+14; Gia+16] and keyword extraction [TMV16; Tix+19]. These contributions, along with several other works conducted by the research community, resulted in a survey article [Mal+20] and various tutorials [MPV16] dedicated to the topic of k-core decomposition. Lastly, my work has explored various aspects of the community detection problem [MV13a; MMV17]. These works further motivated part of the methodological contributions presented in this manuscript.

1.1.2 *Structure-aware Random Walk Node Embeddings*

In the early stages of graph representation learning, random walk-based embedding models gained considerable attention, mainly due to the flexible way of determining node proximity in a stochastic manner. Two of the most widely-used random walk models, namely DeepWalk [PARS14] and Node2Vec [GL16], first sample a set of truncated random walks for each node on the graph to define center-context node relationships; the embeddings are then optimized to maximize the likelihood of node co-occurrences within the random walks. In collaboration primarily with my Ph.D. student Abdulkadir Çelikkanat, we have proposed various expressive and scalable random walk embedding models. These contributions are discussed in Chapter 2.

First, we have investigated how to learn expressive node embeddings by leveraging the underlying graph's community structure. We have proposed Topical Node Embeddings (TNE), a framework in which node embeddings are enhanced with topic (or community) information towards learning topic-aware node representations—leading to performance improvements on downstream tasks [ÇM21].

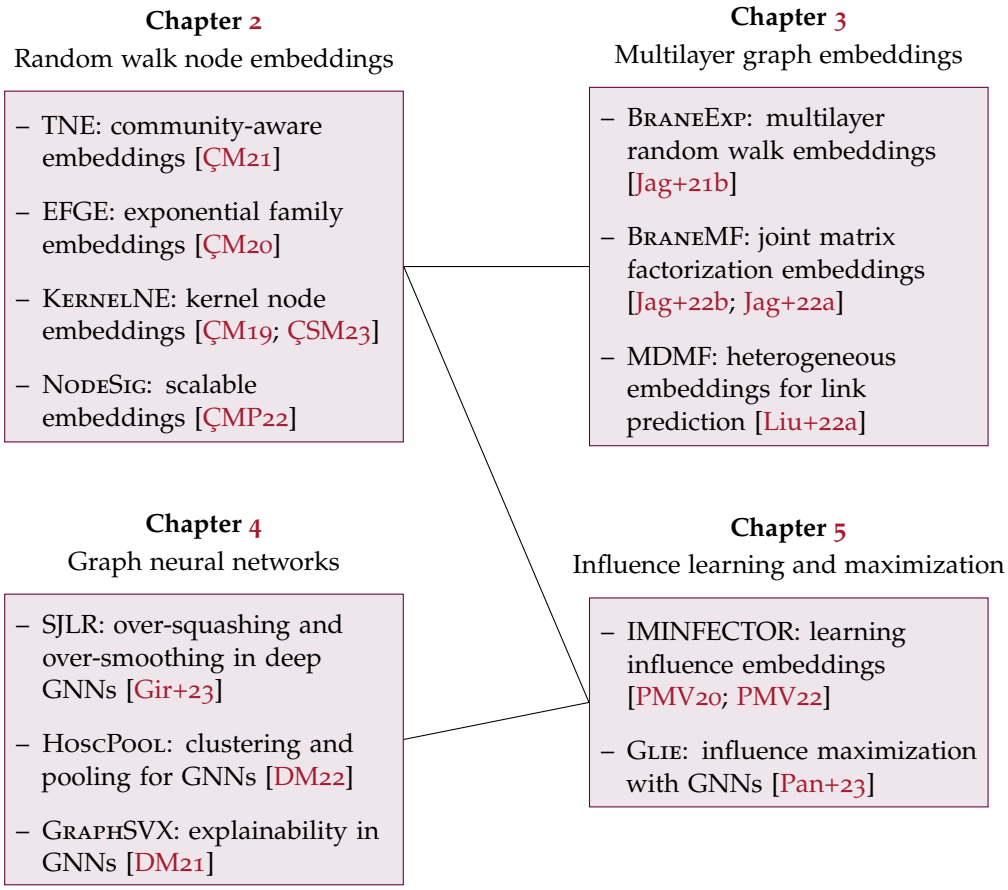


Figure 1.1: Illustration of the work presented in this manuscript and dependencies between the different chapters.

Subsequently, to improve model flexibility, we have employed expressive conditional probability models to relate nodes within random walk sequences. We have introduced Exponential Family Graph Embeddings (EFGE), a family of models that generalize random walk representation learning techniques to exponential family conditional distributions [ÇM20].

Third, we have studied methodologies that combine matrix factorization with random walks for graph representation learning. In collaboration with Yanning Shen (UC Irvine), we have introduced a kernelized weighted matrix factorization model called Kernel Node Embeddings (KernelNE). The model has further been extended to a multiple kernel learning formulation [ÇM19; ÇSM23].

Lastly, we have focused on the efficiency aspects of random walk node embeddings. Sampling random walks and maximizing the likelihood of node co-occurrences could be a computational bottleneck in analyzing large-scale graphs. To address this challenge, in collaboration with Apostolos N. Papadopoulos (Aristotle Univ. of Thessaloniki), we have introduced NodeSig that combines random walk diffusion probabilities with random projection hashing to compute binary node embeddings [ÇMP22].

1.1.3 *Multilayer Graph Embeddings*

A multilayer graph is an elegant model in graph theory to represent networks with multiple types of relationships between nodes [Boc+14; Kiv+14]. Motivated by data

integration problems in computational biology, we have investigated methodologies to learn embeddings on multilayer graphs. This line of work is presented in Chapter 3.

In collaboration with my Ph.D. student Surabhi Jagtap and researchers from IFP Energies nouvelles (a research institution specializing in the energy and environmental sectors), we have introduced representation learning models that rely on random walks. Our first such proposal, BraneExp, combines a biased random walk sampling procedure with the models presented in Chapter 2 [Jag+21b]. Furthermore, to avoid explicitly sampling random walk sequences from a multilayer graph – a time-consuming step that requires extensive tuning – we have leveraged random walk diffusion probabilities. This has resulted in the formulation of BraneMF, a joint matrix factorization framework for multilayer graphs [Jag+22b; Jag+22a]. The performance of the models has been evaluated on various downstream prediction tasks for biological data analysis.

We have also explored methods for learning embeddings on multi-relational heterogeneous graphs, particularly in the context of drug-target interaction (DTI) prediction in drug development. In collaboration with Bin Liu, Dimitrios Papadopoulos, Grigorios Tsoumakas, and Apostolos N. Papadopoulos (Aristotle Univ. of Thessaloniki), we have introduced MDMF, a supervised link prediction framework capable of leveraging heterogeneous information in the context of a multilayer graph [Liu+22a].

1.1.4 *Topics in Graph Neural Networks*

Graph Neural Networks (GNNs) have recently emerged as an attempt to extend neural networks and the deep learning paradigm to graph data [Ham20; Bro+21]. We have delved into various aspects of GNNs, ranging from design issues to explainability. These contributions are presented in Chapter 4.

Most current GNN architectures suffer from performance degradation due to the over-smoothing and over-squashing effects when multiple layers are stacked to form deep models. In collaboration with Jhony H. Giraldo, a visiting Ph.D. student from La Rochelle Université (currently Assistant Professor at Télécom Paris), Konstantinos Skianis (BLUAI), and Thierry Bouwmans (La Rochelle Université), we have identified a fundamental trade-off between over-smoothing and over-squashing stemming from the topological properties of the graph. We have also presented SJLR, an algorithm based on graph curvature designed to alleviate both phenomena [Gir+23].

The second aspect concerns the design of pooling operators for GNNs used to compute graph-level representations (e.g., for graph classification tasks). Given that global pooling discards graph structure during the computation of its final representation, our focus has been on developing a structure-aware graph pooling model. In this direction, with my Ph.D. student Alexandre Duval, we have proposed HoscPool, a clustering-based pooling operator based on a motif spectral clustering formulation [DM22].

Lastly, we have considered the problem of explainability in GNNs toward allowing end-users to understand model predictions. With Alexandre, we have proposed GraphSVX, an explainability model utilizing Shapley values to highlight the significance of both node features and graph structure in the predictions [DM21].

1.1.5 *Graph Representations for Influence Learning and Maximization*

Modeling online information and influence spread constitute critical tasks in computational social science and beyond. To this end, detecting influential spreaders, i.e., individuals who can disseminate information effectively, is crucial for increasing our un-

derstanding of spreading processes within a networked system. Instances of the problem can be found in the core of various practical applications, ranging from computational epidemiology to media polarization and misinformation detection. The problem has been studied extensively at its combinatorial optimization formulation, aka the influence maximization problem, focusing primarily on improving time complexity [Li+18]. In collaboration with my Ph.D. student George Panagopoulos, along with Nikolaos Tziortziotis (Jellyfish) and Michalis Vazirgiannis (École Polytechnique), we have investigated how to leverage graph representation learning in the tasks of influence learning and optimization. This topic is covered in Chapter 5.

First, we have examined how real information cascades (e.g., retweets in X, formerly Twitter) can be used for influence learning and maximization. We have introduced IMINFECTOR (Influence Maximization with Influencer Vectors), a scalable model that leverages embeddings learned from diffusion cascades, enabling model-independent influence maximization [PMV20; PMV22].

Then, we have studied the more general case where ground truth information cascades are unavailable. Specifically, we have proposed the GLIE model (Graph Learning-based Influence Estimation) that performs influence estimation relying on GNNs to encode the structural position of the nodes in the graph. We have further shown how to use the model for influence maximization [Pan+23].

STRUCTURE OF THE MANUSCRIPT. This manuscript is structured along four main chapters (Chapter 2 to 5) devoted to the different lines of research presented above. To keep the manuscript self-contained, each chapter begins with a concise presentation of background concepts. Chapter 6 concludes the manuscript with perspectives about ongoing and future research directions. Finally, Appendix A provides a complete list of publications and Appendix B a curriculum vitae.

2

LEARNING GRAPH EMBEDDINGS WITH RANDOM WALKS

This chapter describes our contributions to learning unsupervised node embeddings with random walks. After providing the necessary background material in Sec. 2.1, we present four models addressing different challenges. The first model, called TNE, aims to leverage the community structure of graphs in the learning process (Sec. 2.2). Then, we discuss EFGE, a technique that improves the expressiveness of random walk models with exponential family distributions (Sec. 2.3). The third approach, KernelNE, investigates how kernel methods can be used to augment the predictive capacity of random walk matrix factorization (Sec. 2.4). Finally, we present NodeSig, a random walk model that computes binary embedding balancing scalability and efficiency (Sec. 2.5). A discussion of the contributions and perspectives is given in Sec. 2.6. The chapter is based on the following publications:

- Abdulkadir Çelikkanat and Fragkiskos D. Malliaros. Topic-Aware Latent Models for Representation Learning on Networks. *Pattern Recognition Letters* 144 (2021), pp. 89–96.
- Abdulkadir Çelikkanat and Fragkiskos D. Malliaros. Exponential Family Graph Embeddings. In *AAAI,*. 2020, pp. 3357–3364.
- Abdulkadir Çelikkanat, Yanning Shen, and Fragkiskos D. Malliaros. Multiple Kernel Representation Learning on Networks. *IEEE Trans. Knowl. Data Eng.* 35:6 (2023), pp. 6113–6125.
- Abdulkadir Çelikkanat and Fragkiskos D. Malliaros. Kernel Node Embeddings. In *GlobalSIP,*. 2019, pp. 1–5.
- Abdulkadir Çelikkanat, Fragkiskos D. Malliaros, and Apostolos N. Papadopoulos. NodeSig: Binary Node Embeddings via Random Walk Diffusion. In *ASONAM,*. 2022, pp. 68–75.

2.1 BACKGROUND

A graph $G = (\mathcal{V}, \mathcal{E})$ consists of a set of nodes (or vertices) $\mathcal{V}$ and a set of edges (or links) $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ which connect node pairs. In this chapter, our focus lies on learning *unsupervised node embeddings* that preserve the structure of the graph and its various properties in low-dimensional space representations. The problem is typically formulated as an optimization task, where proximity relationships in the learned space reflect the structure and properties of the original graph [Ham20; HYL17b; MT21]. Let $G = (\mathcal{V}, \mathcal{E})$ be a graph, (M, d) be a metric space, and $\mathbf{S}(u, v), \forall u, v \in \mathcal{V}$ be a matrix encoding a graph-based pairwise similarity function. We aim to find a mapping $\mathbf{E} : \mathcal{V} \to \mathsf{M}$ minimizing the following reconstruction loss:

$$\sum_{(v,u)\in\mathcal{D}} \ell\Big(d_{\mathsf{M}}\big(\mathbf{E}[u], \mathbf{E}[v]\big), \mathbf{S}(u, v)\Big). \tag{2.1}$$

Here, $\ell(\cdot,\cdot)$ represents a loss function that quantifies the difference between the estimated similarity value $d_{\mathsf{M}}(\mathbf{E}[u],\mathbf{E}[v])$ and the actual similarity $\mathbf{S}(u,v)$. The set $\mathcal{D}$ refers to the training set. Typically, the embedding space M is chosen to be $\mathbb{R}^d$, i.e., $\mathbf{E}[v] \in \mathbb{R}^d$ for each node $v \in \mathcal{V}$, $d(\cdot,\cdot) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, and $\ell(\cdot,\cdot) : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$.

We can think of the above formulation as an encoder-decoder architecture. Specifically, the *encoder* will map each node $v \in \mathcal{V}$ into a low-dimensional embedding, i.e., $\textsc{Encoder}(v) = \mathbf{E}[v] \in \mathbb{R}^d$. The decoder aims to reconstruct node similarities $\mathbf{S}(u,v)$ between nodes u and v based on the embeddings $\mathbf{E}[u]$ and $\mathbf{E}[v]$, i.e., $\textsc{Decoder}(\mathbf{E}[u],\mathbf{E}[v]) \approx \mathbf{S}(u,v)$ [HYL17b]. In this chapter, we focus on node embedding techniques that involve simple encoder functions, commonly known as shallow embeddings in the literature. This formulation will help us later on to introduce more complex neural encoders in Chapter 4.

A common way of formulating this structure-preserving embedding learning process is through the lens of matrix factorization, aiming to obtain a low-dimensional representation of similarity matrix $\mathbf{S}$. Some of the earliest approaches here include the widely used IsoMap [TSL01] and Laplacian Eigenmaps [BN01] algorithms. More recently, matrix factorization node embedding models have been introduced, targeting both first- and higher-order proximity information [Ahm+13; CLX15; Ou+16; WCZ16; Ros+20]. In most cases, Singular Value Decomposition (SVD) is applied to obtain a low-rank approximation of the proximity matrix corresponding to the embeddings. Nevertheless, these models suffer from high time complexity. Besides, they mainly rely on user-specified, hand-designed node proximity criteria, which can also be challenging to compute on large-scale graphs.

To improve flexibility, random walk-based methods have gained considerable attention, mainly due to their ability to determine node proximity in a stochastic manner. These methods aim at modeling *center-context* node relationships, maximizing the probability of node co-occurrences in random walk sequences. Let $\mathbf{w} = (v_1,\ldots,v_l,\ldots,v_{\mathcal{L}}) \in \mathcal{V}^{\mathcal{L}}$ be a walk (i.e., sequence of nodes) of length $\mathcal{L}$ extracted from a graph, where each $v_l \in \mathcal{V}$ for all $l \in \{1,\ldots,\mathcal{L}\}$. Considering a center node v_l, the surrounding nodes within a certain distance γ, $v_{l-\gamma},\ldots,v_{l-1}$, $v_{l+1},\ldots,v_{l+\gamma}$, correspond to its context. Then, node embeddings can be learned by optimizing the following objective function:

$$\mathcal{F}_N(\mathbf{A},\mathbf{B}) := \underset{\Omega}{\operatorname{argmax}} \sum_{\mathbf{w}\in\mathcal{W}} \sum_{l=1}^{\mathcal{L}} \sum_{\substack{-\gamma\le j\le\gamma \\ j\neq 0}} \log\ \Pr(v_{l+j} \mid v_l;\ \Omega), \tag{2.2}$$

where $\Omega = (\mathbf{A},\mathbf{B})$ is the model parameters that we would like to learn, $\mathcal{W}$ is the set of random walks $\mathbf{w} = (v_1,\ldots,v_l,\ldots,v_{\mathcal{L}}) \in \mathcal{V}^{\mathcal{L}}$ of length $\mathcal{L}$, and γ refers to the window size. Following suit of the SkipGram model for word embeddings [Mik+13a; Mik+13b], $\mathbf{A}[v]$ and $\mathbf{B}[v]$ correspond to the embedding vectors when v is considered as the context and center node, respectively. Typically, the *softmax* function is used for the probability measure in (2.2) defined as

$$\Pr(v_{l+j}|v_l) := \frac{\exp\left(\mathbf{A}[v_{l+j}]^\top \cdot \mathbf{B}[v_l]\right)}{\sum_{v\in\mathcal{V}} \exp\left(\mathbf{A}[v]^\top \cdot \mathbf{B}[v_l]\right)}. \tag{2.3}$$

This idea was first introduced in the DeepWalk model which considers uniform random walks [PARS14]. It was then extended by Node2Vec to biased random walks, enabling the sampling of nodes that interpolate between breadth-first search and depth-first search explorations. Regarding model optimization, *hierarchical softmax* or *negative sampling*

are employed to deal with the computational challenges imposed while evaluating the denominator of (2.3).

2.2 COMMUNITY-AWARE NODE EMBEDDINGS

> This section is based on material from a journal article [ÇM21] co-authored with Abdulkadir Çelikkanat, published in *Pattern Recognition Letters*.

While random walk-based embedding models outlined in Sec. 2.1 effectively capture local connectivity patterns, they often fall short in conveying comprehensive information about the global structural properties of the graph. Specifically, real-world networks exhibit inherent clustering or community structure, which can be harnessed to enhance the predictive capabilities of node embeddings. One can interpret such structural information based on an analogy to the concept of *topics* in a collection of documents. Similar to how word embeddings can be augmented with topic-based information [Liu+15], our objective here is to strengthen node embeddings by incorporating knowledge about the latent community structure of the graph. This can be achieved through a process akin to topic modeling.

In this section, we present a framework in which node embeddings are enhanced with topic (or community) information towards learning topic-aware embeddings—something that leads to further improvements in the performance of downstream tasks. Specifically, we first show how existing latent space discovery models, such as community detection and topic models, can be incorporated into the node representation learning process. Then, we introduce *Topical Node Embeddings* (TNE), a model that follows a two-step approach: firstly, learning community embeddings from the graph and subsequently utilizing them to enhance the node representations derived from random walk methods. We investigate different variations of this model and analyze their properties.

2.2.1 *Learning Topic Representations*

Complex networks, such as those arising from social or biological settings, consist of latent clusters of different sizes in which the nodes are more likely to be connected to each other [GN02; For10; MV13a]. Our main goal here is to use the latent clusters of a network to obtain enriched representations. This can be achieved by enhancing node embedding vectors with *topic representations*. By replacing a node v_i with its community label z_i in a random walk, we learn *community embeddings* by predicting the nodes in the context of a community label. More formally, we can define our objective function to learn topic embeddings as follows:

$$\mathcal{F}_T(\widetilde{\mathbf{A}}, \widetilde{\mathbf{B}}) := \underset{\widetilde{\Omega}=(\widetilde{\mathbf{A}},\widetilde{\mathbf{B}})}{\operatorname{argmax}} \sum_{\mathbf{w}\in\mathcal{W}} \sum_{l=1}^{\mathcal{L}} \sum_{\substack{-\gamma\le j\le\gamma \\ j\neq 0}} \log \Pr(v_{l+j} \mid z_l;\, \widetilde{\Omega}). \tag{2.4}$$

By maximizing the log-probability of (2.4), we obtain the embedding vectors corresponding to each community label $z_i \in \{1, \ldots, \mathcal{K}\}$, where $\mathcal{K}$ indicates the number of latent communities. In this work, we mainly use two approaches to detect latent communities. The first one is based on a novel combination of generative statistical models accompanied by random walks, while the second one relies on traditional community detection algorithms that leverage the inherent structure of the graph.

RANDOM WALKS AND GENERATIVE GRAPH MODELS. Most real-world networks can be expressed as a combination of nested or overlapping communities [Pal+05]. Therefore, when a random walk is sampled, it not only visits neighboring nodes but also traverses communities in the network (see Fig. 2.1 (b)). In that regard, we assume that each random walk can be represented as random mixtures over latent communities, and each community can be characterized by a distribution over nodes. In other words, we can write the following generative model for each walk over the network:

1. For each $k \in \{1, \ldots, \mathcal{K}\}$
   - $\phi_k \sim \text{Dirichlet}(b_0)$
2. For each walk $\mathbf{w} = (v_1, \ldots, v_l, \ldots, v_L)$
   - $\theta_{\mathbf{w}} \sim \text{Dirichlet}(a_0)$
   - For each vertex $v_l \in \mathbf{w}$
     - $z_i \sim \text{Multinomial}(\theta_{\mathbf{w}})$
     - $v_l \sim \text{Multinomial}(\phi_{z_l})$

Here, $\mathcal{N}$ is the number of walks and $\mathcal{L}$ is the walk length. Considering each random walk as a document and the collection of random walks as a corpus, it can be seen that the statistical process defined above corresponds to the well-known Latent Dirichlet Allocation (LDA) model [BNJ03]. We will refer to this model as GLDA. As we show in Lemma 2.1, the relative frequency of the occurrences of a node in the generated walks is proportional to its degree in the network for a large number of walks or walk lengths. This property was first empirically demonstrated in the work by Perozzi *et al*. [PARS14], allowing SKIPGRAM-based models to be applied to real-world graphs. Here, we provide a formal argument for this empirical observation.

Lemma 2.1. *Let $G = (\mathcal{V}, \mathcal{E})$ be a connected graph, and $\{X_l\}_{l \geq 1}$ be a Markov chain with state space $\mathcal{V} = \{1, \ldots, N\}$ and transition matrix $\mathbf{P}$, where $P(v, u)$ is defined as $1/deg(v)$ for each edge $(v, u) \in \mathcal{E}$ and 0 otherwise. If the Markov chain is aperiodic, then*

$$\lim_{\mathcal{L} \to \infty} \frac{1}{L} \mathbb{E}\left[O_{\mathcal{L}}^{(v)}\right] = \frac{deg(v)}{2|\mathcal{E}|},$$

where $O_{\mathcal{L}}^{(v)}$ is a random variable representing the number of occurrences of the node v in a random walk of length $\mathcal{L}$.

A variant of the GLDA model has also been introduced to incorporate the hidden state of the current node when determining the next vertex to be visited. This model, denoted by GHMM, is based on the Hidden Markov Model (HMM) with symmetric Dirichlet priors over transition and emission distributions.

GRAPH STRUCTURE-BASED MODELING. The previous models have relied on the generated random walks to detect each node's community (or topic) assignment in the given node sequence. To increase the flexibility of the TNE framework, we utilize two additional community detection models, which directly target extracting communities of nodes from a given graph. The first model corresponds to the well-known LOUVAIN algorithm that extracts communities based on modularity maximization [Blo+08], while the second one to the BIGCLAM model for detecting overlapping communities [YL13].

2.2.2 *Topical Nodel Embeddings*

This section outlines the Topical Node Embedding (TNE) model, which has been proposed to learn topic-aware node representations. Figure 2.1 gives an overview of the model. The

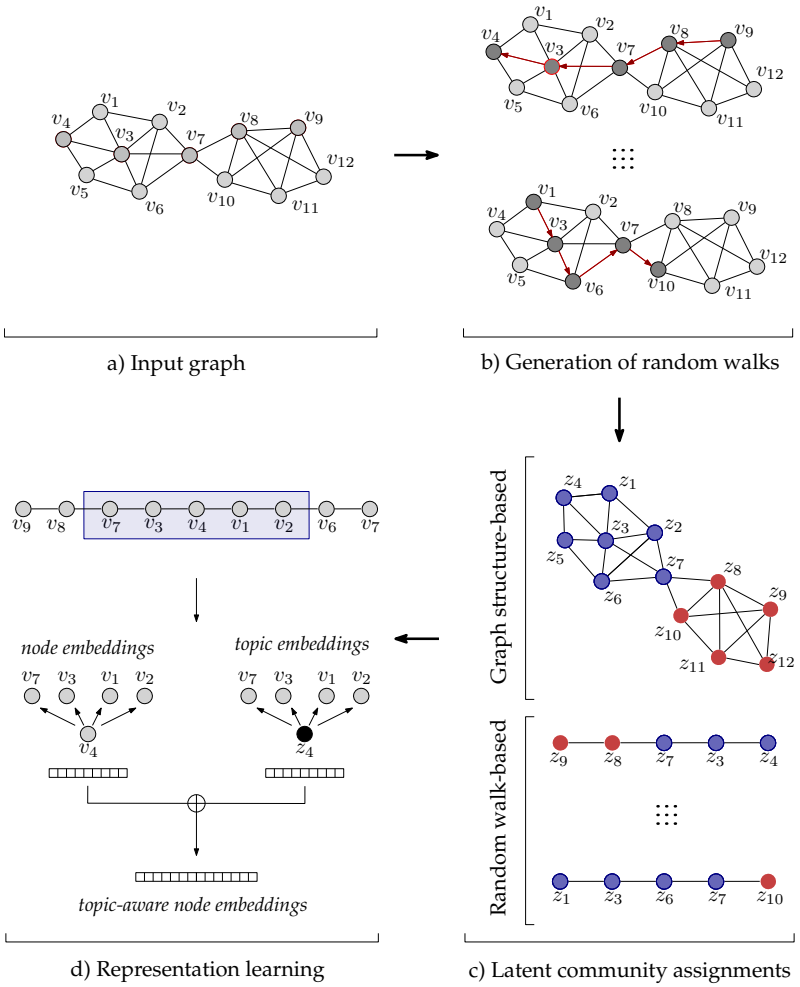


Figure 2.1: Schematic representation of the TNE model. The final representations are learned by combining node and topic embeddings. The representation of a node is learned using random walks performed over the network; its topic representation is similarly learned by assigning a topic/community label based either on random walks (TNE-GLDA, TNE-GHMM) or network structure-based approaches (TNE-LOUVAIN, TNE-BIGCLAM).

objective of TNE is to enhance node embeddings using information about the underlying topics of the graph obtained by the models described in Sec. 2.2.1. This can be achieved by learning node and topic embedding vectors independently of each other, jointly maximizing the objectives defined in Equations (2.2) and (2.4):

$$\underset{\Omega,\widetilde{\Omega}}{\operatorname{argmax}} \sum_{\mathbf{w}\in\mathcal{W}} \sum_{l=1}^{\mathcal{L}} \sum_{\substack{-\gamma\le j\le\gamma \\ j\neq 0}} \Big[\underbrace{\log \Pr\left(v_{l+j}|v_l;\Omega\right)}_{\text{node embeddings}} + \underbrace{\log \Pr\left(v_{l+j}|z_l;\widetilde{\Omega}\right)}_{\text{community embeddings}} \Big].$$

After obtaining the node and topic representations, our final step is to efficiently incorporate these two feature vectors, $\mathbf{B}[v]$ and $\widetilde{\mathbf{B}}[z]$ of node v and community label z respectively, to obtain the final topic-enhanced node embedding. For this purpose, we

Algorithm 2.1 Topical Node Embeddings (TNE)

Input: Graph $G = (\mathcal{V}, \mathcal{E})$; number of walks, $\mathcal{N}$; walk length, $\mathcal{L}$; window size, γ; number of communities, $\mathcal{K}$; topic representation learning method, T; node embedding size, d_n; community embedding size, d_t

Output: Embedding vectors of length $d_n + d_t$

1: Perform $\mathcal{N}$ random walks of length $\mathcal{L}$ for each node
2: Learn node embeddings by optimizing (2.2)
3: Learn topic embeddings by optimizing (2.4), using any of the models T of Sec. 2.2.1
4: Concatenate node and topic embeddings with (2.5)

concatenate node embedding vector $\mathbf{B}[v]$ with the expected topic vector with respect to the distribution $\Pr(\cdot|v)$. Our strategy can be formulated as follows:

$$\mathbf{B}[v] \oplus \sum_{k \in \{1,\dots,\mathcal{K}\}} \Pr(k|v) \cdot \widetilde{\mathbf{B}}[k], \tag{2.5}$$

where $\oplus$ indicates the concatenation operation. We refer to the final vector obtained after concatenating the node and topic feature vectors as *topical node embedding*. Algorithm 2.1 provides the pseudocode of the proposed TNE model. Depending on the method used to learn topical representations, we will refer to the corresponding instances of TNE as TNE-GLDA, TNE-GHMM, TNE-LOUVAIN, and TNE-BIGCLAM. The different instances of the TNE model are empirically evaluated in Sec. 2.4.2, along with the models presented in the following two sections.

2.3 EXPONENTIAL FAMILY NODE EMBEDDINGS

This section is based on material from an article co-authored with Abdulkadir Çelikkanat published in the *AAAI Conference on Artificial Intelligence (AAAI)* [ÇM20].

As mentioned in Sec. 2.1, random walk-based approaches commonly generate sets of node sequences from the input graph. The key distinction between these methods lies in their approach to generating (i.e., sampling) sequences. They create center-context node pairs by examining the occurrences of nodes within a certain distance of each other within the walks. Subsequently, popular NLP models like the SKIPGRAM model [Mik+13b] are employed to learn latent node embeddings.

Nevertheless, SKIP-GRAM models the conditional distribution of nodes within a random walk using the *softmax* function, which can limit its ability to capture more complex interaction patterns between nodes that co-occur in a random walk. Acknowledging this limitation of existing random walk-based methods, we argue that employing more expressive *conditional probability models* to relate nodes within a random walk sequence could yield more informative latent node representations.

In this section, we leverage *exponential family distribution* models to capture node interactions in random walks. Exponential families offer a mathematically convenient and flexible parametric set of probability distributions to represent relationships between entities. Specifically, we introduce the *Exponential Family Graph Embeddings* (EFGE) model, which extends random walk techniques to exponential family conditional distributions. We investigate three specific instances of the EFGE model, corresponding to well-known exponential family distributions: Bernoulli, Poisson, and Normal. Furthermore, we ana-

lyze the objective function of the proposed parametric models, providing connections to well-known unsupervised graph learning models under appropriate parameter settings.

2.3.1 *Overview of Exponential Families*

Before presenting the proposed model, let us briefly introduce exponential family distributions, a parametric set of probability distributions that include, among others, the Gaussian, Bernoulli, and Poisson distributions. A class of probability distributions is referred to as exponential family distributions when they can be expressed in the following form:

$$p(y) = h(y) \exp\Big(\eta T(y) - A(\eta)\Big), \tag{2.6}$$

where h is the *base measure*, η are the *natural parameters*, T is the *sufficient statistic* of the distribution, and $A(\eta)$ is the *log-normalizer* or *log-partition* function [And70]. Through the selection of distinct base measures and sufficient statistics, we can obtain different probability distributions. For instance, the base measure and sufficient statistic of the Bernoulli distribution are $h(y) = 1$ and $T(y) = y$, respectively, while for the Poisson distribution we have $h(y) = 1/y!$ and $T(y) = y$.

As mentioned above, exponential families contain a wide range of commonly used distributions, providing a general class of models by re-parameterizing distributions in the natural parameters η. That way, we will use the natural parameter η to design a set of node embedding models, defining $\eta_{v,u}$ as the product of context and center vectors in the following way:

$$\eta_{v,u} := f\Big(\mathbf{A}[v]^\top \cdot \mathbf{B}[u]\Big),$$

where f is called the *link function*. As we will present shortly in the following section, we have many alternative options for the form of the link function $f(\cdot)$.

2.3.2 *Learning Node Embeddings with Exponential Families*

This section introduces the proposed family of EFGE models. The main idea is to leverage the expressive capability of exponential family distributions to condition context nodes based on the center node of interest. First, we will describe the formulation of the general objective function of the EFGE model, and then we will present particular instances of the model.

Let $\mathcal{W}$ denote a collection of node sequences generated using a random walk strategy applied to a given graph G. Based on that, we can define a generic objective function to learn node embeddings in the following way:

$$\mathcal{F}(\mathbf{A}, \mathbf{B}) := \underset{\Omega}{\operatorname{argmax}} \sum_{\mathbf{w} \in \mathcal{W}} \sum_{l=1}^{\mathcal{L}} \sum_{v \in \mathcal{V}} \log p\Big(y^{(l)}_{(\mathbf{w},v)}; \Omega\Big), \tag{2.7}$$

where $y^{(l)}_{(\mathbf{w},v)}$ is the observed value indicating the relationship between the center w_l and context node v. Note that the objective function in (2.7) is quite similar to the one of the SkipGram model presented in (2.2), except that we also include nodes that are not belonging to context sets. Rather than limiting ourselves to the *sigmoid* or *softmax* functions for modeling the probability in the objective function of (2.7), we introduce a

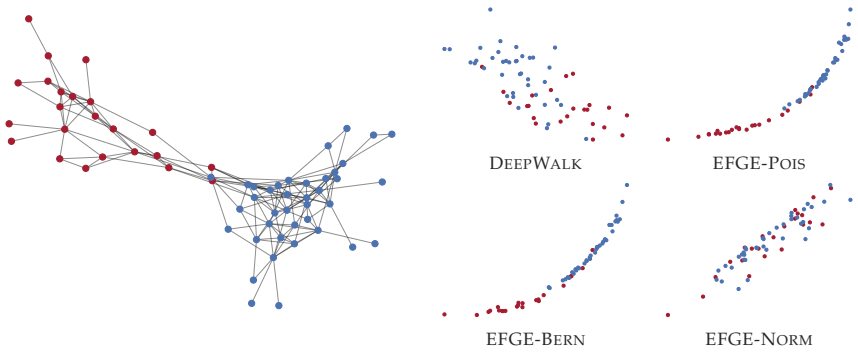


Figure 2.2: Visualization of the *Dolphins* network composed by two communities and the corresponding embeddings for $d = 2$.

generalization that assumes each $y^{(l)}_{\mathbf{w},v}$ follows an exponential family distribution. That way, the objective function used to learn node embedding vectors $\Omega = (\mathbf{A}, \mathbf{B})$ can be rewritten as follows:

$$\underset{\Omega}{\operatorname{argmax}} \sum_{\mathbf{w}\in\mathcal{W}} \sum_{l=1}^{\mathcal{L}} \sum_{v\in\mathcal{V}} \log h(y^{(l)}_{\mathbf{w},v}) + \eta_{w_l,v} T(y^{(l)}_{\mathbf{w},v}) - A(\eta_{w_l,v}). \tag{2.8}$$

We have examined three instances of the EFGE model that represent well-known exponential family distributions. Specifically, we employ the Bernoulli, Poisson, and Normal distributions leading to the corresponding EFGE-BERN, EFGE-POIS, and EFGE-NORM models. For illustration purposes, Fig. 2.2 depicts the *Dolphins* network composed by two communities and the embeddings in two dimensions as computed by different models. As we can see, in this specific toy example, the proposed EFGE-BERN and EFGE-POIS models learn representations that can distinguish nodes based on their communities. In the following sections, we analyze the properties of these models.

THE EFGE-BERN MODEL. Our first model is EFGE-BERN, in which node occurrences within the context set of another mode follow a Bernoulli distribution. Let $Y^{(l)}_{\mathbf{w},v}$ be a random variable following a Bernoulli distribution which is equal to 1 if node v appears in the context set $\mathcal{C}^{(l)}_{\gamma}(\mathbf{w})$ of node w_l. We can express it as $Y^{(l)}_{(\mathbf{w},v)} = X^{(l-\gamma)}_{(\mathbf{w},v)} \vee \cdots \vee X^{(l-1)}_{(\mathbf{w},v)} \vee X^{(l+1)}_{(\mathbf{w},v)} \vee \cdots \vee X^{(l+\gamma)}_{(\mathbf{w},v)}$, where each $X^{(l+j)}_{(\mathbf{w},v)}$ indicates the appearance of v at the specific position $l + j$ $(-\gamma \leq j \neq 0 \leq \gamma)$ in the walk $\mathbf{w} \in \mathcal{W}$. Here, we assume that $X^{(l+j)}_{(\mathbf{w},v)}$'s are independent variables. We can express the objective function of the EFGE-BERN model, $\mathcal{F}_B(\mathbf{A}, \mathbf{B})$, by splitting (2.8) into two parts with respect to the values of $Y_{(\mathbf{w},v)}$ and $X^{(l+j)}_{(\mathbf{w},v)}$:

$$\begin{aligned}\mathcal{F}_B(\mathbf{A}, \mathbf{B}) &= \underset{\Omega}{\operatorname{argmax}} \sum_{\mathbf{w}\in\mathcal{W}} \sum_{l=1}^{\mathcal{L}} \left(\sum_{v\in\mathcal{C}^{(l)}_{\gamma}(\mathbf{w})} \log p\left(y^{(l)}_{(\mathbf{w},v)}\right) + \sum_{v\notin\mathcal{C}^{(l)}_{\gamma}(\mathbf{w})} \log p\left(y^{(l)}_{(\mathbf{w},v)}\right) \right) \\ &= \underset{\Omega}{\operatorname{argmax}} \sum_{\mathbf{w}\in\mathcal{W}} \sum_{l=1}^{\mathcal{L}} \sum_{\substack{-\gamma\leq j\leq\gamma \\ j\neq 0}} \left(\log p\left(x^{(l+j)}_{(\mathbf{w},w_{l+j})}\right) + \sum_{\substack{v\in\mathcal{V} \\ v\neq w_{(l+j)}}} \log p\left(x^{(l+j)}_{(\mathbf{w},v)}\right) \right).\end{aligned} \tag{2.9}$$

Note that the exponential form of a Bernoulli distribution with a parameter π is equal to $\exp\left(\eta x - A(\eta)\right)$, where the log-normalizer $A(\eta)$ is $\log(1+\exp(\eta))$ and the parameter π is the sigmoid function $\sigma(\eta) = 1/\left(1-\exp(-\eta)\right)$. Therefore, we can rewrite the objective function $\mathcal{F}_B(\mathbf{A},\mathbf{B})$ as follows:

$$\mathcal{F}_B(\mathbf{A},\mathbf{B}) = \underset{\Omega}{\operatorname{argmax}} \sum_{\mathbf{w}\in\mathcal{W}} \sum_{l=1}^{\mathcal{L}} \sum_{\substack{-\gamma\leq j\leq\gamma \\ j\neq 0}} \left(\log\sigma\Big(\eta_{(w_l,w_{l+j})}\Big) + \sum_{\substack{v\in\mathcal{V} \\ v\neq w_{l+j}}} \log\sigma\Big(-\eta_{(w_l,v)}\Big)\right). \tag{2.10}$$

We choose the identity map for the link function $f(\cdot)$, so $\eta_{w_l,v}$ directly becomes equal to the product of vectors $\mathbf{A}[v]$ and $\mathbf{B}[w_l]$. Finally, in Lemma 2.2, we show that the log-likelihood $\mathcal{F}_B(\mathbf{A},\mathbf{B})$ of the EFGE-BERN model in fact converges to the objective function of negative sampling given in (2.11).

Lemma 2.2. *The log-likelihood function $\mathcal{L}_B(\mathbf{A},\mathbf{B})$ can be approximated by*

$$\sum_{w\in\mathcal{W}} \sum_{l=1}^{\mathcal{L}} \sum_{\substack{-\gamma\leq j\leq\gamma \\ j\neq 0}} \left(\log p\Big(x^{(l+j)}_{(w,w_{l+j})}\Big) + \sum_{r=1}^{k} \mathop{\mathbb{E}}_{s\sim q^-} \Big[\log p\Big(x^{(l+j)}_{(w,s)}\Big)\Big]\right) \tag{2.11}$$

for large values of k.

THE EFGE-POIS MODEL. This model employs the Poisson distribution to capture the relationship between context and center nodes in a random walk sequence. Let $Y^{(l)}_{(\mathbf{w},v)}$ be a random variable indicating the number of occurrences of node v in the context of w_l. We assume that $Y^{(l)}_{(\mathbf{w},v)}$ follows a Poisson distribution, with the mean value $\widetilde{\lambda}_{(w_l,v)}$ being the number of appearances of node v in the context of w_l within the window size γ. Similar to the previous model, it can be expressed as $Y^{(l)}_{(\mathbf{w},v)} = X^{(l-\gamma)}_{(\mathbf{w},v)} + \cdots + X^{(l-1)}_{(\mathbf{w},v)} + X^{(l+1)}_{(\mathbf{w},v)} + \cdots + X^{(l+\gamma)}_{(\mathbf{w},v)}$, where $X^{(l+j)}_{(\mathbf{w},v)} \sim \text{Poisson}\Big(\lambda^{(l+j)}_{(w_l,v)}\Big)$ for $-\gamma \leq j \neq 0 \leq \gamma$. That way, we obtain $\widetilde{\lambda}_{(w_l,v)} = \sum_{j=-\gamma}^{\gamma} \lambda^{(l+j)}_{(w_l,v)}$, since the sum of independent Poisson random variables is also Poisson. By substituting the exponential form of the Poisson distribution into (2.7), we can obtain the objective function of the model as:

$$\mathcal{F}_P(\mathbf{A},\mathbf{B}) = \underset{\Omega}{\operatorname{argmax}} \sum_{\mathbf{w}\in\mathcal{W}} \sum_{l=1}^{\mathcal{L}} \sum_{v\in\mathcal{V}} \left(\log h\Big(y^{(l)}_{(\mathbf{w},v)}\Big) + \Big(\widetilde{\eta}_{(w_l,v)} y^{(l)}_{(\mathbf{w},v)} - \exp\left(\widetilde{\eta}_{(w_l,v)}\right)\Big)\right),$$

where the base measure $h\Big(y^{(l)}_{(\mathbf{w},v)}\Big)$ is equal to $1/y^{(l)}_{(\mathbf{w},v)}!$. Note that, the number of occurrence $y^{(l)}_{(\mathbf{w},v)}$ is equal to 0 if v does not appear in the context of $w_l \in \mathcal{V}$. Using a similar approach as in the EFGE-BERN model, the equation can be divided into two parts based on the cases where $y_{(\mathbf{w},v)} > 0$ and $y_{(\mathbf{w},v)} = 0$. That way, we can apply negative sampling (given in (2.11)) as follows:

$$\sum_{\mathbf{w}\in\mathcal{W}} \sum_{l=1}^{L} \sum_{\substack{-\gamma\leq j\leq\gamma \\ j\neq 0}} \Bigg(\Big(-\log\left(x^{(l+j)}_{(\mathbf{w},w_{l+j})}!\right) + \eta_{(w_l,w_{l+j})} x^{(l+j)}_{(\mathbf{w},w_{l+j})} - \exp\left(\eta_{(w_l,w_{l+j})}\right)\Big) +$$

$$\sum_{r=1}^{k} \mathop{\mathbb{E}}_{s\sim q^-} \Big[-\exp\left(\eta_{(w_l,s)}\right) \Big] \Bigg).$$

Note that, in the EFGE-Pois model, we do not specify any particular link function; thus, the natural parameter is equal to the product of the embedding vectors.

THE EFGE-NORM MODEL. If a node v appears more frequently in the context of w_l compared to other nodes, we can infer that v exhibits a higher level of interaction with w_l than the other nodes. Therefore, we consider each $y^{(l)}_{(\mathbf{w},v)}$ in this model as a weight indicating the strength of the relationship between nodes w_l and v. We assume that $X^{(l+j)}_{(\mathbf{w},v)} \sim \mathcal{N}(1,\sigma_+^2)$ if $v \in \mathcal{C}^{(l)}_\gamma(\mathbf{w})$, and $X^{(l+j)}_{(\mathbf{w},v)} \sim \mathcal{N}(0,\sigma_-^2)$ otherwise. Hence, we can conclude that $Y^{(l)}_{(\mathbf{w},v)} \sim \mathcal{N}(\widetilde{\mu},\widetilde{\sigma}^2)$ if we follow a similar assumption $Y^{(l)}_{(\mathbf{w},v)} = \sum_{j=-\gamma}^{\gamma} X^{(l+j)}_{(\mathbf{w},v)}$ as in the previous model, where $\widetilde{\mu}$ is the number of occurrences of v in the context. The formulation of the objective function of the EFGE-Norm model is given as follows:

$$\mathcal{F}_N(\mathbf{A},\mathbf{B}) = \sum_{\mathbf{w}\in\mathcal{W}} \sum_{l=1}^{\mathcal{L}} \sum_{\substack{-\gamma\le j\le\gamma \\ j\neq 0}} \Bigg(\Bigg(\log h\Big(x^{(l+j)}_{(\mathbf{w},w_{l+j})}\Big) + \Big(x^{(l+j)}_{(\mathbf{w},w_{l+j})} \frac{\eta_{(w_l,w_{l+j})}}{\sigma^+} - \frac{\eta^2_{(w_l,w_{l+j})}}{2}\Big)\Bigg)$$
$$+ \sum_{r=1}^{k} \mathop{\mathbb{E}}_{s\sim q^-} \left[\log h\Big(x^{(l+j)}_{(\mathbf{w},s)}\Big) + \Big(x^{(l+j)}_{(\mathbf{w},s)} \frac{\eta_{(w_l,s)}}{\sigma^-} - \frac{\eta^2_{(w_l,s)}}{2}\Big) \right] \Bigg),$$

where the base measure $h(x)$ is $\exp(-x^2/2\sigma^2)/\sqrt{2\pi}\sigma$ for known variance. In this model, we set the link function to $f(x) = \exp(-x)$, so $\eta_{(w_l,v)}$ is defined as $\exp(-\mathbf{A}[v]^\top \cdot \mathbf{B}[w_l])$.

2.4 KERNEL NODE EMBEDDINGS

This section is based on material from an article co-authored with Abdulkadir Çelikkanat and Yanning Shen, published in *IEEE Trans. Knowl. Data Eng. (TKDE)* [ÇSM23]. A preliminary version of this work was published in the *IEEE Global Conference on Signal and Information Processing (GlobalSIP)* [ÇM19].

So far, we have introduced graph representation learning models that learn embeddings by maximizing the likelihood of nodes co-occurring within a set of random walk sequences. As we briefly discussed in Sec. 2.1, another way of learning node representations is from a matrix factorization perspective. In this case, a matrix capturing node similarities is factorized to learn a low-dimensional approximation using SVD [CLX15; Ou+16; Wan+17]. Here, we study how to bring together the best of both worlds—random walks and matrix factorization—through a *kernel method*.

Kernel functions have often been introduced along with popular learning algorithms, such as PCA [SSM97], SVM [SS01], Spectral Clustering [DGK04], and Collaborative Filtering [Liu+16a], to name a few. Traditional learning models often struggle to capture the underlying substructures present in complex datasets due to their reliance on linear techniques, which are inadequate for modeling nonlinear patterns in the data. Kernel functions [HSS08], on the other hand, allow mapping nonlinearly separable points into a (generally) higher dimensional feature space so that the inner product in the new space can be computed without needing to compute the exact feature maps—bringing further computational benefits. Additionally, to further reduce model bias, *multiple*

kernel learning approaches have been proposed to learn optimal combinations of kernel functions [GA11].

Kernel functions have also been utilized in the field of graph analysis. At the node level, diffusion kernels and their applications [KL02] constitute notable instances. At the graph level, *graph kernels* [Vis+10], such as the random walk and Weisfeiler-Lehman kernels [KJM20], have mainly been utilized to measure the similarity between a pair of graphs for applications such as graph classification. Relevant approaches have also been proposed to leverage graph kernels for node classification in a supervised manner [Tia+19] and in semi-supervised link prediction [BBS11].

In this section, we introduce KERNELNE (*Kernel Node Embeddings*), an unsupervised model that combines matrix factorization and random walks within a kernelized framework for learning node embeddings. The potential benefit of this modeling approach lies in its ability to harness and blend the elegant mathematical formulation provided by matrix factorization with the expressive capabilities of random walks to capture a notion of "stochastic" node similarity in an efficient way. More importantly, this formulation enables leveraging kernel functions in the node embedding learning task. Due to the inherent characteristics of matrix factorization models, node similarities can be perceived as inner products of vectors lying in a latent space—which allows employing kernels to interpret the embeddings in a higher dimensional feature space using non-linear maps. To further improve expressiveness, we introduce MKERNELNE, a multiple kernel learning formulation of the model. It extends the kernelized weighted matrix factorization framework by learning a linear combination of a predefined set of kernels.

2.4.1 *Kernel-based Representation Learning*

The general objective function for our problem is defined as a weighted matrix factorization [SJ03], expressed as follows:

$$\mathcal{F}(\mathbf{A},\mathbf{B}) := \underset{\mathbf{A},\mathbf{B}}{\operatorname{argmin}} \underbrace{\left\|\mathbf{W}\odot\left(\mathbf{M}-\mathbf{A}\mathbf{B}^\top\right)\right\|_F^2}_{\text{Error term}} + \underbrace{\frac{\lambda}{2}\left(\|\mathbf{A}\|_F^2+\|\mathbf{B}\|_F^2\right)}_{\text{Regularization term, } \mathcal{R}(\mathbf{A},\mathbf{B})}. \tag{2.12}$$

$\mathbf{M}\in\mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ represents the target matrix constructed from the desired properties of a given graph, which is used to learn node embeddings $\mathbf{A},\mathbf{B}\in\mathbb{R}^{|\mathcal{V}|\times d}$. Each element $\mathbf{W}(v,u)$ of the weight matrix $\mathbf{W}\in\mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ captures the importance of the approximation error between nodes v and u, and $\odot$ indicates the Hadamard product. Depending on the desired graph properties that we are interested in encoding, there are many possible alternatives to choose matrix $\mathbf{M}$; such include the number of common neighbors between a pair of nodes, higher-order node proximity based on the *Adamic-Adar* or *Katz* indices [Ou+16], as well leveraging multi-hop information [CLX15]. Here, we will design $\mathbf{M}$ as a sparse binary matrix utilizing information of random walks over the network.

Consider a collection of walks of length $\mathcal{L}$ denoted as $\mathcal{W}$, and let γ represent the window size. Let $\mathbf{M}(v,u)$ be a binary value that equals 1 if node u appears in the context of v in any random walk. Also, let $\mathbf{F}(v,u)$ be $2\cdot\gamma\cdot\#(v)$, where $\#(v)$ indicates the total number of occurrences of node v in the generated walks. Setting each term $\mathbf{W}(v,u)$ as the square root of $\mathbf{F}(v,u)$, the objective function in (2.12) can be expressed under a random walk-based formulation as follows:

$$\mathcal{F}(\mathbf{A},\mathbf{B}) = \underset{\mathbf{A},\mathbf{B}}{\operatorname{argmin}} \left\|\sqrt{\mathbf{F}}\odot\left(\mathbf{M}-\mathbf{A}\mathbf{B}^\top\right)\right\|_F^2 + \mathcal{R}(\mathbf{A},\mathbf{B})$$

$$= \underset{\mathbf{A},\mathbf{B}}{\operatorname{argmin}} \sum_{v\in\mathcal{V}} \sum_{u\in\mathcal{V}} \mathbf{F}(v,u)\Big(\mathbf{M}(v,u) - \big\langle \mathbf{A}[u], \mathbf{B}[v]\big\rangle\Big)^2 + \mathcal{R}(\mathbf{A},\mathbf{B})$$

$$= \underset{\mathbf{A},\mathbf{B}}{\operatorname{argmin}} \sum_{\mathbf{w}\in\mathcal{W}} \sum_{l=1}^{\mathcal{L}} \sum_{\substack{j=-\gamma \\ j\neq 0}}^{\gamma} \sum_{u\in\mathcal{V}} \Big(\mathbb{1}_{\{w_{l+j}\}}(u) - \big\langle \mathbf{A}[u], \mathbf{B}[w_l]\big\rangle\Big)^2 + \mathcal{R}(\mathbf{A},\mathbf{B}). \tag{2.13}$$

Matrix $\mathbf{A}$ in (2.13) represents the embedding vectors of nodes when they are treated as *centers*. These embeddings are utilized for the downstream prediction tasks.

Before moving forward, let us consider the case where the entries of the weight matrix $\mathbf{W}$ in (2.12) are equal to one. It is well-known that the low-rank matrix that minimizes the unweighted sum-squared distance to matrix $\mathbf{M}$ is precisely determined by the leading components of the SVD of $\mathbf{M}$. Nevertheless, in our case, the weight matrix is not uniform, making it challenging to directly apply SVD without the precise realization of the target matrix. To overcome this limitation, we employ kernel functions to learn node representations through matrix factorization.

Let (X, d_X) be a metric space and $\mathbb{H}$ be a Hilbert space of real-valued functions defined on X. A Hilbert space is called *reproducing kernel Hilbert space* (RKHS) if the point evaluation map over $\mathbb{H}$ is a continuous linear functional [HSSo8]. Furthermore, a *feature map* is defined as a function $\Phi : \mathsf{X} \to \mathbb{H}$ from the input space X into *feature space* $\mathbb{H}$. Every feature map defines a *kernel* $\mathsf{K} : \mathsf{X} \times \mathsf{X} \to \mathbb{R}$ as follows:

$$\mathsf{K}(\mathbf{x},\mathbf{y}) := \langle \Phi(\mathbf{x}), \Phi(\mathbf{y})\rangle \qquad\qquad \forall (\mathbf{x},\mathbf{y}) \in \mathsf{X}^2.$$

It can be seen that $\mathsf{K}(\cdot,\cdot)$ is symmetric and positive definite due to the properties of an inner product space.

A function $g : \mathsf{X} \to \mathbb{R}$ is *induced by* K, if there exists $h \in \mathbb{H}$ such that $g = \langle h, \Phi(\cdot)\rangle$ for a feature vector Φ of kernel K. Note that, this is independent of the definition of the feature map Φ and space $\mathbb{H}$ [Steo2]. Let $\mathcal{I}_\kappa := \{g : \mathsf{X} \to \mathbb{R} \mid \exists h \in \mathbb{H} \text{ s.t. } g = \langle h, \Phi(\cdot)\rangle\}$ be the set of induced functions by kernel K. Then, a continuous kernel K on a compact metric space $(\mathsf{X}, d_\mathsf{X})$ is *universal*, if the set $\mathcal{I}_\kappa$ is dense in the space of all continuous real-valued functions $\mathcal{C}(\mathsf{X})$. In other words, for any function $f \in \mathcal{C}(\mathsf{X})$ and $\epsilon > 0$, there exists $g_h \in \mathcal{I}_\kappa$ satisfying

$$\|f - g_h\|_\infty \leq \epsilon,$$

where g_h is defined as $\langle h, \Phi(\cdot)\rangle$ for some $h \in \mathbb{H}$. In our approach, we consider universal kernels, since we can always find $h \in \mathbb{H}$ satisfying $|\langle h, \phi(x_i)\rangle - \alpha_i| \leq \epsilon$ for given $\{x_1, \ldots, x_N\} \subset \mathsf{X}$, $\{\alpha_1, \ldots, \alpha_N\} \subset \mathbb{R}$ and $\epsilon > 0$ [Steo2]. If we choose α_i's as the entries of a row of our target matrix $\mathbf{M}$, then the elements h and $\phi(x_i)$ indicate the corresponding row vectors of $\mathbf{A}$ and $\mathbf{B}$, respectively. Then, we can decompose the target matrix by repeating the process for each row.

SINGLE KERNEL NODE EMBEDDING LEARNING. Following the discussion above, we can perform matrix factorization in the feature space using kernel functions. In particular, we can transfer the inner product operation from the input space X to the feature space $\mathbb{H}$ by reformulating (2.13) in the following manner:

$$\mathcal{F}_S := \underset{\mathbf{A},\mathbf{B}}{\operatorname{argmin}} \sum_{\mathbf{w}\in\mathcal{W}} \sum_{l=1}^{\mathcal{L}} \sum_{\substack{j=-\gamma \\ j\neq 0}}^{\gamma} \sum_{u\in\mathcal{V}} \Big(\mathbb{1}_{\{w_{l+j}\}}(u) - \big\langle \Phi(\mathbf{A}[u]), \Phi(\mathbf{B}[w_l])\big\rangle\Big)^2 + \mathcal{R}(\mathbf{A},\mathbf{B})$$

$$= \underset{\mathbf{A},\mathbf{B}}{\operatorname{argmin}} \sum_{\mathbf{w}\in\mathcal{W}} \sum_{l=1}^{\mathcal{L}} \sum_{\substack{j=-\gamma \\ j\neq 0}}^{\gamma} \sum_{u\in\mathcal{V}} \left(\mathbb{1}_{\{w_{l+j}\}}(u) - \mathsf{K}\big(\mathbf{A}[u], \mathbf{B}[w_l]\big) \right)^2 + \mathcal{R}(\mathbf{A}, \mathbf{B}). \tag{2.14}$$

Specifically, we use the following two universal kernel functions:

$$\mathsf{K}_G(\mathbf{x}, \mathbf{y}) = \exp\left(\frac{-\|\mathbf{x}-\mathbf{y}\|^2}{\sigma^2} \right) \qquad \sigma \in \mathbb{R}$$

$$\mathsf{K}_S(\mathbf{x}, \mathbf{y}) = \frac{1}{\left(1 + \|\mathbf{x}-\mathbf{y}\|^2\right)^{\sigma}} \qquad \sigma \in \mathbb{R}_+,$$

where K_G and K_S correspond to the *Gaussian* and *Schoenberg* kernels, respectively. We will refer to the proposed kernel-based node embedding methodology as KernelNE (the two different kernels will be denoted by Gauss and Sch).

Regarding model optimization, for each center node $w_l \in \mathcal{V}$ in (2.14), we have to compute the gradient for each $u \in \mathcal{V}$, which is computationally intractable. However, note that (2.14) can be divided into two parts concerning the values of $\mathbb{1}_{\{w_{l+j}\}}(u) \in \{0, 1\}$. Thus, negative sampling can be employed, where for each center node w_l, we sample k negative instances s^- from the noise distribution p^- as follows:

$$\mathcal{F}_S := \underset{\mathbf{A},\mathbf{B}}{\operatorname{argmin}} \sum_{\mathbf{w}\in\mathcal{W}} \sum_{l=1}^{\mathcal{L}} \sum_{\substack{j=-\gamma \\ j\neq 0}}^{\gamma} \Bigg(\big(1 - \mathsf{K}\big(\mathbf{A}[w_{l+j}], \mathbf{B}[w_l]\big)^2 + \sum_{\substack{r=1 \\ s_r^- \sim p^-}}^{k} \mathsf{K}\big(\mathbf{A}[s_r^-], \mathbf{B}[w_l]\big)^2 \Bigg) + \mathcal{R}(\mathbf{A}, \mathbf{B}). \tag{2.15}$$

MULTIPLE KERNEL NODE EMBEDDING LEARNING. Selecting a proper kernel function $\mathsf{K}(\cdot, \cdot)$ and the corresponding parameters (e.g., the bandwidth of a Gaussian kernel) is a critical task during the learning phase. However, relying on a single kernel function might introduce inherent bias and restrict the performance of the model. The ability to effectively employ multiple kernels can enhance model expressiveness, enabling the capturing of diverse notions of similarity among embeddings [GA11]. Besides, learning how to combine such kernels might further improve the performance of the underlying model. In particular, given a set of base kernels $\{\mathsf{K}_i\}_{i=1}^K$, we aim to find an optimal way to combine them, as follows:

$$\mathsf{K}^{\mathbf{c}}(\mathbf{x}, \mathbf{y}) = f_{\mathbf{c}}\big(\{\mathsf{K}_i(\mathbf{x}, \mathbf{y})\}_{i=1}^K | \mathbf{c}\big),$$

where the combination function $f_{\mathbf{c}}$ is parameterized on $\mathbf{c} \in \mathbb{R}^K$ that indicates kernel weights. Due to the generality of the multiple kernel learning framework, $f_{\mathbf{c}}$ can be either a linear or nonlinear function. In our study, we have explored how to enhance the proposed kernelized weighted matrix factorization model by linearly combining multiple kernels. Let $\mathsf{K}_1, \ldots, \mathsf{K}_K$ be a set of kernel functions satisfying the properties presented in

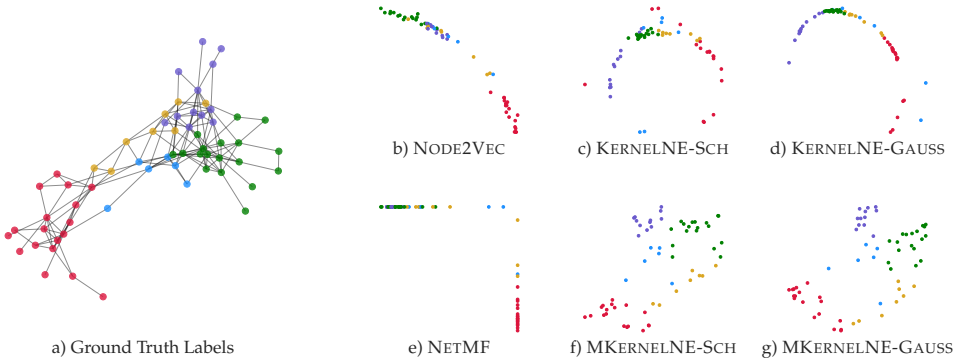


Figure 2.3: Visualization of embeddings of the *Dolphins* network. Node colors indicate community labels computed by the LOUVAIN algorithm.

the previous paragraph. Then, we can reformulate the objective function in the following manner:

$$\mathcal{F}_M := \underset{\mathbf{A},\mathbf{B},\mathbf{c}}{\operatorname{argmin}} \sum_{\mathbf{w}\in\mathcal{W}} \sum_{l=1}^{\mathcal{L}} \sum_{\substack{j=-\gamma \\ j\neq 0}}^{\gamma} \Bigg(\Big(1 - \sum_{i=1}^{K} c_i \mathsf{K}_i(\mathbf{A}[w_{l+j}], \mathbf{B}[w_l])\Big)^2 + \sum_{\substack{r=1 \\ s_r^- \sim p^-}}^{k} \Big(\sum_{i=1}^{K} c_i \mathsf{K}_i(\mathbf{A}[s_r^-], \mathbf{B}[w_l])\Big)^2 \Bigg) + \frac{\lambda}{2}\left(\|\mathbf{A}\|_F^2 + \|\mathbf{B}\|_F^2\right) + \frac{\beta}{2}\|\mathbf{c}\|_2^2, \tag{2.16}$$

where $\mathbf{c} = [c_1, \ldots, c_K]^\top \in \mathbb{R}^K$. Here, we introduce an additional parameter c_j representing the contribution of the corresponding kernel K_i. $\beta > 0$ is a tradeoff parameter for the regularization term, and similarly, the coefficients $c_1, \ldots, c_K$ are optimized by fixing the remaining model parameters $\mathbf{A}$ and $\mathbf{B}$. Equation (2.16) corresponds to the objective function of the proposed multiple kernel learning model MKERNELNE.

Finally, Fig. 2.3 shows a simple visualization of the embeddings of the *Dolphins* network computed by different variants of KERNELNE and MKERNELNE as well as two baseline models. We have used the LOUVAIN algorithm [Blo+08] to detect the communities in the network. As we can observe, different instances of MKERNELNE learn embeddings in which nodes of the different communities are better distributed in the two-dimensional space. A clustering experiment on the embedding vectors has further supported this observation, where the multiple kernel models achieve higher normalized mutual information (NMI) scores.

2.4.2 *Experimental Evaluation of TNE, EFGE, and (M)KernelNE*

In this section, we briefly evaluate the performance of the models introduced in the previous sections, namely TNE, EFGE, and (M)KERNELNE, in the tasks of node classification and link prediction. We have decided to present jointly an overview of the empirical analysis since all these models aim to learn node embeddings in an unsupervised manner. A detailed list of experiments can be found in the corresponding articles for TNE [ÇM21], EFGE [ÇM20], and (M)KERNELNE [ÇSM23].

DATASETS. We have conducted experiments on real-world datasets, including the *CiteSeer* and *Cora* citation networks; the *DBLP*, *AstroPh*, *HepTh* collaboration networks; a *PPI* (protein-protein interaction) network; a snapshot of the *Facebook* social graph; and the *Gnutella* peer-to-peer file-sharing network [LK14].

BASELINE MODELS. In our analysis, we have considered several baseline models, including DEEPWALK [PARS14], NODE2VEC [GL16], LINE [Tan+15], VERSE [Tsi+18], HOPE [Ou+16], NETMF [Qiu+18b], GEMSEC [Roz+19], and M-MNF [Wan+17]. We apply the same parameter values for the random walk approaches, with 80 walks per node, the walk length set to 10, and the window size set to 10. To keep the presentation short and coherent, we only report the performance of the best-performing baseline and the performance of the proposed models.

NODE CLASSIFICATION. In the node classification task, a portion of the network's nodes (training set) is provided with labels, and the objective is to predict the labels of the remaining nodes (test set). After learning the embedding vector for each node, we split them into varying sizes of training and test sets, ranging from 1% up to 90%. We conduct experiments using a one-vs-rest logistic regression classifier with L_2 regularization [Ped+11]. Here, we report the Micro-F_1 scores for a subset of train and test splits (10%, 50%, and 90%), averaged over 50 runs for each representation learning method. As we can observe in Tables 2.1, 2.2, and 2.3, different instances of the EFGE model are quite competitive, especially for smaller training set sizes. Comparing the different instances of the TNE model, as described in Sec. 2.2, we have observed an overall better performance of the graph structure-based models TNE-LOUVAIN and TNE-BIGCLAM, that can potentially be attributed to the way in which latent communities are extracted. While TNE-GLDA and TNE-GHMM employ random walks for both node and topic representations, it appears that these random walks may not capture the clustering structure as effectively as algorithms specifically designed for this purpose, such as BIGCLAM and LOUVAIN. We have further observed that employing multiple kernels with MKERNELNE frequently yields better performance than the single kernel model, particularly for smaller training ratios.

		TNE				EFGE			KERNELNE		MKERNELNE	
	*Baseline**	*GLDA*	*GHMM*	*LOUVAIN*	*BIGCLAM*	*BERN*	*POIS*	*NORM*	*GAUSS*	*SCH*	*GAUSS*	*SCH*
Citeseer	0.554	0.535	0.526	0.551	0.546	0.565	**0.571**	0.564	0.555	0.561	0.566	<u>0.570</u>
Cora	0.773	0.751	0.762	0.765	0.754	**0.787**	0.784	<u>0.786</u>	0.780	0.765	0.781	0.772
DBLP	0.623	0.617	0.619	0.625	0.625	<u>0.639</u>	**0.640**	**0.640**	0.617	0.627	0.636	0.631
PPI	0.164	0.173	0.172	0.178	0.180	0.179	0.184	0.185	0.148	**0.195**	**0.195**	<u>0.194</u>

*Best-performing baseline. *Citeseer*: VERSE; *Cora*: NETMF; *DBLP*: NODE2VEC; *PPI*: NODE2VEC. Boldface and underline: best and second-best performing model.

Table 2.1: Comparison of models for node classification with 10% training ratio.

LINK PREDICTION. In the link prediction task, we remove half of the edges in the network while preserving its connectivity. Subsequently, we learn node embeddings on the residual network. The edges that have been removed are considered positive

		TNE				EFGE			KernelNE		MKernelNE	
	*Baseline**	*GLDA*	*GHMM*	*LOUVAIN*	*BIGCLAM*	*BERN*	*POIS*	*NORM*	*GAUSS*	*SCH*	*GAUSS*	*SCH*
Citeseer	0.595	0.610	0.583	0.619	0.612	**0.621**	**0.621**	0.617	0.611	0.602	0.613	0.609
Cora	0.834	**0.840**	0.824	0.835	0.828	0.835	0.822	0.829	0.837	0.800	0.823	0.806
DBLP	0.639	0.634	0.635	0.642	0.641	0.655	0.655	**0.656**	0.633	0.637	0.648	0.641
PPI	0.226	0.233	0.227	0.226	0.229	0.227	0.231	0.234	0.192	**0.244**	0.242	0.241

*Best-performing baseline. *Citeseer*: Node2Vec; *Cora*: NetMF; *DBLP*: Node2Vec; *PPI*: LINE.

Table 2.2: Comparison of models for node classification with 50% training ratio.

		TNE				EFGE			KernelNE		MKernelNE	
	*Baseline**	*GLDA*	*GHMM*	*LOUVAIN*	*BIGCLAM*	*BERN*	*POIS*	*NORM*	*GAUSS*	*SCH*	*GAUSS*	*SCH*
Citeseer	0.620	0.618	0.594	**0.638**	0.624	0.626	0.627	0.625	0.620	0.609	0.620	0.615
Cora	0.845	**0.856**	0.834	0.850	0.851	0.847	0.834	0.837	0.851	0.812	0.833	0.818
DBLP	0.641	0.638	0.634	0.642	0.642	0.656	**0.658**	**0.658**	0.636	0.640	0.650	0.643
PPI	0.242	0.251	0.238	0.240	0.244	0.240	0.242	0.247	0.203	**0.256**	0.254	0.251

*Best-performing baseline. *Citeseer*: VERSE; *Cora*: Node2Vec; *DBLP*: Node2Vec; *PPI*: LINE.

Table 2.3: Comparison of models for node classification with 90% training ratio.

samples for the testing set. The same number of node pairs that do not exist in the original graph is sampled randomly to form the negative samples. Then, the entries of the feature vector corresponding to each node pair (u, v) in the test set are computed based on the element-wise operation $|\mathbf{A}(v, i) - \mathbf{A}(u, i)|^2$, for each coordinate axis i of the embedding vectors $\mathbf{A}(u, :)$ and $\mathbf{A}(v, :)$. In the experiments, we use logistic regression with L_2 regularization. Table 2.4 gives the Area Under Curve (AUC) scores for the link prediction task. A general observation is that the two instances of the kernelized matrix factorization framework, namely KernelNE and MKernelNE, demonstrate the best performance across five datasets. However, it is worth noting that simpler baseline models like DeepWalk and Node2Vec often exhibit comparable levels of accuracy.

RUNNING TIME. To compare the scalability of the different models, we have generated Erdös-Rényi $\mathcal{G}_{n,p}$ graphs [New10] of varying sizes, ranging from 2^8 to 2^{13} nodes. In the generation of random graphs, we set the edge probabilities so that the expected node degree is 10. Figure 2.4 shows the running time of the different variants of TNE and KernelNE. Regarding TNE, as expected, TNE-Louvain is the most scalable instance due to the efficient way of inferring the community structure. KernelNE and MKernelNE are more time-efficient models, having running times comparable to DeepWalk and Node2Vec. For instance, to compute the embeddings of an Erdös-Rényi graph with 10^5 nodes, DeepWalk requires 9,525 seconds, Node2Vec 1,684, while KernelNE and MKernelNE 3,845 and 6,486 seconds, respectively. Besides, considering multiple kernels does not significantly affect the scalability properties of the MKernelNE model. EFGE also performs similarly, thus omitted from the comparison. Nevertheless, the models studied so far cannot easily scale to large graphs mainly (i) because they have to sample

		TNE				EFGE			KernelNE		MKernelNE	
	Baseline*	GLDA	GHMM	LOUVAIN	BIGCLAM	BERN	POIS	NORM	GAUSS	SCH	GAUSS	SCH
Citeseer	0.828	0.809	0.793	0.809	0.795	0.820	0.829	0.783	0.807	**0.882**	0.850	0.863
Cora	0.781	0.775	0.806	0.780	0.767	0.753	0.777	0.782	0.774	**0.823**	0.802	0.810
DBLP	0.944	0.958	0.959	0.959	0.958	0.954	0.957	0.956	0.950	0.960	**0.961**	0.960
PPI	**0.886**	0.772	0.872	0.780	0.844	0.728	0.739	0.806	0.846	0.801	0.768	0.788
AstroPh	0.969	0.977	**0.979**	0.977	0.977	0.977	0.973	0.966	0.963	0.970	**0.979**	0.970
HepTh	0.896	0.903	0.908	0.905	0.904	0.899	0.902	0.907	0.899	**0.917**	**0.917**	0.916
Facebook	0.984	**0.993**	**0.993**	**0.993**	**0.993**	0.991	0.991	0.990	0.990	0.988	0.989	0.989
Gnutella	**0.839**	0.728	0.796	0.721	0.723	0.622	0.624	0.668	0.816	0.673	0.653	0.671

*Best-performing baseline. *Citeseer*: DeepWalk; *Cora*: Node2Vec; *DBLP*: DeepWalk, Node2Vec; *PPI*: M-NMF; *AstroPh*: LINE; *HepTh*: DeepWalk, Node2Vec; *Facebook*: DeepWalk; *Gnutella*: LINE.

Table 2.4: Comparison of models in the link prediction task.

a large set of random sequences explicitly; and (ii) due to the computationally expensive optimization procedures required to learn embeddings [Zha+18; LGP21]. To overcome these challenges, we introduce a scalable node embedding methodology in Sec. 2.5.

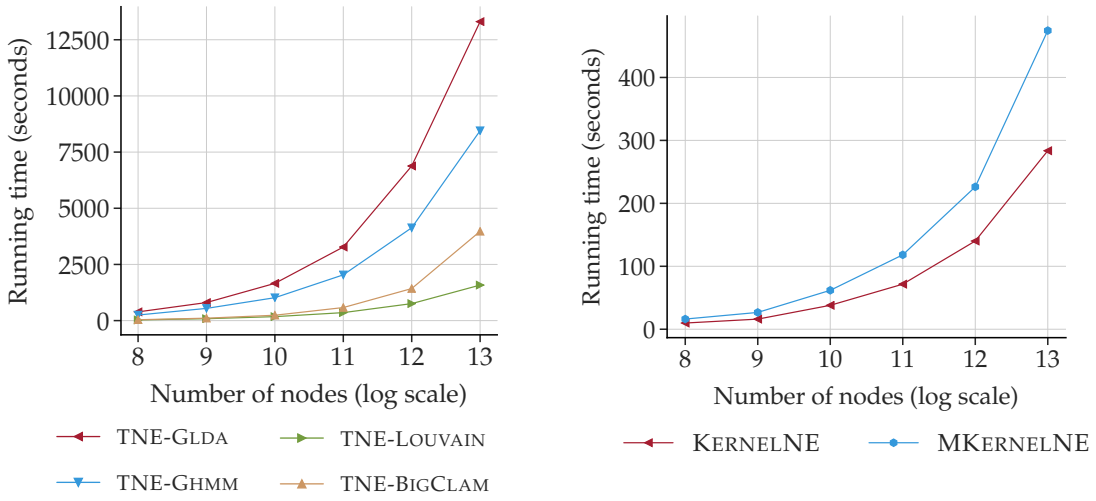


Figure 2.4: Running time of different models on Erdös-Rényi random graphs of different sizes.

2.5 SCALABLE NODE EMBEDDINGS

This section is based on material from an article co-authored with Abdulkadir Çelikkanat and Apostolos N. Papadopoulos, published in the *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)* [ÇMP22].

The vast majority of node embedding techniques, including those presented in Sections 2.2, 2.3, and 2.4 deal with *learning-based* models. In particular, they rely on matrix factorization or random walk sampling to infer and optimize the proximities between nodes

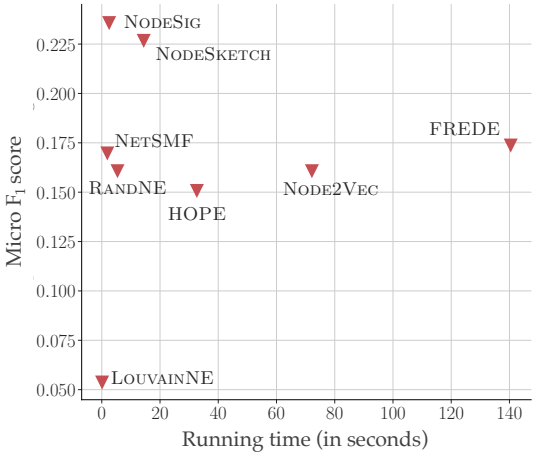


Figure 2.5: Comparison of models (running time vs. accuracy) on the *PPI* graph ($|\mathcal{V}| = 4K$, $|\mathcal{E}| = 40K$).

[Ham20]. Such approaches have certain limitations. First, even if the Skip-Gram model can be optimized relatively efficiently, a large number of random walks is required to be explicitly sampled to ensure the effectiveness of the embeddings on downstream tasks. In the case of matrix factorization [CLX15; Ou+16], the heavy reliance on dense node proximity matrices poses computational and memory limitations, especially when dealing with large-scale networks. Despite recent efforts to enhance the running time complexity through dimensionality reduction [Zha+18; Che+19; Tsi+21], matrix sparsification techniques [Qiu+19], and harnessing hierarchical graph representations [Che+18; Bho+20; LGP21], there is often a trade-off between computational cost and the quality of embeddings. This dilemma presents a challenge for practitioners who must choose between effectiveness and computational efficiency. In addition to the computational challenges in model optimization, most of the proposed algorithms focus on learning low-dimensional embeddings in Euclidean space. A few recent studies have introduced the concept of learning *discrete* node representations [Lia+18; Yan+19], where the similarity between embedding vectors is determined using the *Hamming* distance. These approaches are built upon fast sketching techniques for scalable similarity search, primarily relying on data-independent or data-dependent hashing [Wan+18]. While binary embeddings speed up distance computations compared to metrics defined in the Euclidean space, the learning procedures for such models often become computationally intensive, particularly in the case of learning-to-hash models [Lia+18].

In this section, we will introduce NodeSig, a scalable model for computing binary node embeddings based on stable random projections. NodeSig first leverages random walk diffusions to estimate higher-order node proximity. Then, a properly defined sign random projection hashing technique is applied to obtain binary node signatures in the Hamming space, leading to an approximation of the *chi similarity* (χ) [PW10] between the proximity vectors in the original space. Since these vectors are constructed based on the occurrence frequencies of nodes within random walks, *chi* similarity emerges as a natural choice of similarity metric, frequently used to compare histograms in various areas, including natural language processing and computer vision. Every component of NodeSig has been designed to ensure scalability without compromising the accuracy of downstream tasks. In fact, NodeSig demonstrates comparable or even improved performance compared to traditional models on these tasks. Figure 2.5 positions NodeSig regarding the accuracy

and running time, providing a comparison to different models on the *PPI* network. As we observe, NodeSig's running time is comparable to that of models that focus solely on scalability (e.g., NodeSketch [Yan+19], RandNE [Zha+18], LouvainNE [Bho+20]), with improved accuracy even higher than Node2Vec [GL16], FREDE [Tsi+21], and HOPE [Ou+16] in this dataset.

2.5.1 *Binary Node Signatures via Random Walks*

NodeSig (*Node Signatures*) relies on sign random projections of a properly designed matrix that captures higher-order proximity between nodes. Firstly, we describe the construction of this target matrix by utilizing random walk diffusion probabilities. Then, we show how binary embeddings (or signatures) are obtained by incorporating non-linear mapping through a simple sign function.

RANDOM WALK DIFFUSION FOR NODE PROXIMITY ESTIMATION. To capture higher-order node proximities, we leverage uniform random walk diffusions. Let $\mathbf{P}$ denote the transition matrix associated with the adjacency matrix of the graph, i.e., $\mathbf{P}(i,j) = \mathbf{A}(i,j)/\sum_{j\in\mathcal{V}}\mathbf{A}(i,j)$. We use a slightly modified version of the transition matrix by adding a self-loop on each node in case it does not exist. Note that the probability of transitioning to the next node in the random walk solely depends on the current node. As a result, node v_j can be reached from node v_i by taking l steps with a probability of $\mathbf{P}^{(l)}(i,j)$, given the existence of a connecting path between them. For a given walk length $\mathcal{L}$, we define matrix $\mathbf{M}$ as

$$\mathbf{M} := \mathbf{P} + \cdots + \mathbf{P}^{(l)} + \cdots + \mathbf{P}^{(\mathcal{L})},$$

where $\mathbf{P}^{(l)}$ indicates the l-order proximity matrix and each entry $\mathbf{M}(v,u)$ in fact specifies the expectation of visiting u starting from node v within $\mathcal{L}$ steps. By introducing an additional parameter β, $\mathbf{M}_\beta$ can be rewritten as follows:

$$\mathbf{M}_\beta := \beta\mathbf{P} + \cdots + \beta^{(l)}\mathbf{P}^{(l)} + \cdots + \beta^{(L)}\mathbf{P}^{(L)}. \tag{2.17}$$

By increasing the length of random walks, higher-order node proximity can be captured, and the influence of each step in the walk can be controlled by the *importance factor* $\beta \in \mathbb{R}^+$. In the following paragraph, we will demonstrate how matrix $\mathbf{M}_\beta$ is properly exploited by a random projection hashing strategy to compute binary node representations efficiently.

LEARNING BINARY NODE EMBEDDINGS. To compute node embeddings, we will rely on random projections to represent data points in a lower dimensional space by preserving the similarity in the original space [Vem01]. Here, we aim at encoding each node into a Hamming space $\mathbb{H}\big(d_{\mathcal{H}}, \{0,1\}^d\big)$; we consider the normalized Hamming distance $d_{\mathcal{H}}$ as the distance metric [YCP15]. The benefit of binary representations is twofold: first, they will allow us to perform efficient distance computation using bitwise operations, and second, they reduce the required disk space to store the data.

Random projections are typically linear mappings; to obtain binary embeddings, non-linear functions are required to perform the discretization step. A common approach is to consider the signs of the values obtained by the *Johnson-Lindenstrauss*(JL) [JLS86] transform. More formally, this can be expressed as

$$h_{\mathbf{W}}(\mathbf{x}) := \operatorname{sign}(\mathbf{x}^\top\mathbf{W}),$$

where $\mathbf{W}$ is the projection matrix whose entries $\mathbf{W}(i,j)$ are independently drawn from a normal distribution and $\text{sign}(\mathbf{x})_j$ is equal to 1 if $\mathbf{x}_j > 0$ and 0 otherwise. Goemans and Williamson [GW95] first introduced this approach as a rounding scheme in approximation algorithms, showing that the probability of obtaining different values for a single-bit quantization is proportional to the angle between vectors. Specifically, for a given pair of vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^N$,

$$\Pr\left[h_{\mathbf{W}}(\mathbf{x})_j \neq h_{\mathbf{W}}(\mathbf{y})_j\right] = \frac{1}{\pi}\cos^{-1}\left(\frac{\mathbf{x}^\top \mathbf{y}}{\|\mathbf{x}\|_2\|\mathbf{y}\|_2}\right), \tag{2.18}$$

where $\mathbf{W}(i,j) \sim \mathcal{N}(0,1)$ for $1 \leq i,j \leq N$. The main idea relies on sampling uniformly distributed random hyperplanes in $\mathbb{R}^d$. Each column of the projection matrix, in fact, defines a hyperplane, and the arc between vectors $\mathbf{x}$ and $\mathbf{y}$ on the unit sphere is intersected if $h_{\mathbf{W}}(\mathbf{x})_i$ and $h_{\mathbf{W}}(\mathbf{y})_i$ take different values. While the signs of random projections provide an approximation of the angle between vectors in the original space, in our specific settings, we aim to preserve a distance metric that can better capture the characteristics of the input data $\mathbf{M}_\beta$. Note that the node proximity matrix $\mathbf{M}_\beta$ consists of non-negative elements computed using the occurrence frequencies of nodes within random walks. Therefore, we focus on estimating distance metrics that compare histogram-type data by properly redesigning the projection matrix. We employ *stable* random projections that aim to preserve chi-square (χ^2) similarity [LSH13]. Specifically, α-stable distributions, for stability parameter $0 < \alpha \leq 2$, with unit scale are used to sample the elements of the projection matrix. Li *et al.* [LSH13] proposed the following upper bound

$$\Pr\left[h_{\mathbf{W}}(\mathbf{x})_j \neq h_{\mathbf{W}}(\mathbf{y})_j\right] \leq \frac{1}{\pi}\cos^{-1}\rho_\alpha \tag{2.19}$$

for non-negative vectors ($\mathbf{x}_i \geq 0, \mathbf{y}_i \geq 0$ for $1 \leq i \leq N$), where ρ_α is defined as

$$\rho_\alpha := \left(\frac{\sum_{i=1} x_i^{\alpha/2} y_i^{\alpha/2}}{\sqrt{\sum_{i=1} x_i^\alpha}\sqrt{\sum_{i=1} y_i^\alpha}}\right)^{2/\alpha}.$$

When $\alpha = 2$, (2.19) reduces to the case of normal random projections given in (2.18), also known as SimHash [Cha02]. For $\alpha = 1$, corresponding to Cauchy random projections, when the vectors are chosen from the $\ell^1(\mathbb{R}^+)$ space (i.e., $\sum_{i=1} x_i = 1$, $\sum_{i=1} y_i = 1$), the χ^2 similarity defined as $\rho_{\chi^2} = \sum_{i=1}(2x_i y_i)/(x_i + y_i)$ is always greater or equal to ρ_1. In our case, matrix $\mathbf{M}_\beta$ is sparse enough for small random walk lengths composed of non-negative entries. Thus, we consider designing the projection matrix $\mathbf{W}$ by sampling its entries from the Cauchy distribution, preserving the chi-square similarity ρ_{χ^2}:

$$\mathbb{P}\left[h_{\mathbf{W}}(\mathbf{x})_j \neq h_{\mathbf{W}}(\mathbf{y})_j\right] \approx \frac{1}{\pi}\cos^{-1}\rho_{\chi^2} \leq \frac{1}{\pi}\cos^{-1}\rho_1. \tag{2.20}$$

To obtain the binary node embeddings $\mathbf{E}(v), \forall v \in \mathcal{V}$, we apply a sign-based discretization of the random projections as follows:

$$\mathbf{E}(v) := \left[\text{sign}\Big(\mathbf{M}_\beta(v,:)\ \mathbf{W}(:,1)\Big), \ldots, \text{sign}\Big(\mathbf{M}_\beta(v,:)\ \mathbf{W}(:,d)\Big)\right].$$

Note also that we can overcome the exact realization of $\mathbf{M}_\beta$ through local recursive update rules. This brings a further computational advantage in NodeSig.

2.5.2 *Experimental Evaluation of NodeSig*

BASELINE MODELS. We have considered models targeting efficiency in addition to widely-used node embedding baselines. NETSMF [Qiu+19] is a sparse matrix factorization algorithm, modeling the pointwise mutual information of node co-occurrences. FREDE [Tsi+21] is a matrix sketching-based approach, while RANDNE [Zha+18] leverages Gaussian random projections to deal with scalability. LOUVAINNE [Bho+20] constructs a hierarchical subgraph structure, aggregating the node representations learned at each level. Finally, NODESKETCH [Yan+19] learns embeddings in the Hamming space, using MINHASH signatures. For all methods, we learn embedding vectors of size 128.

NODE CLASSIFICATION. The experiments are conducted by training a one-vs-rest SVM classifier with a pre-computed kernel designed by computing the similarities of node embeddings [Ped+11]. Here, we consider two additional datasets; *Blogcatalog* is a moderate-size social graph with 10K nodes and 340K edges, while *Youtube* is a larger dataset with 1.1M nodes and 3M edges [LK14]. Tables 2.5, 2.6, and 2.7 report the Micro-F_1 scores over ten runs for different training ratios. NODESIG, along with NODESKETCH, another data-independent hashing technique, achieve a competitive performance gain compared to more traditional models like NODE2VEC, especially for larger training ratios.

	HOPE	NODE2VEC	NETSMF	FREDE	LOUVAINNE	RANDNE	NODESKETCH	NODESIG
BlogCat	0.305	0.341	**0.360**	0.354	0.047	0.316	0.305	0.358
Cora	0.687	0.764	0.763	**0.777**	0.686	0.583	0.648	0.750
DBLP	0.620	0.621	0.626	0.648	0.494	0.418	0.668	**0.704**
PPI	0.134	0.141	0.150	0.156	0.042	0.145	0.152	**0.177**
Youtube	0.342	–	0.392	–	0.248	0.335	0.439	**0.455**

Symbol '–' indicates that the model could not run (memory or time constraints).

Table 2.5: Comparison of models for node classification with 10% training ratio.

	HOPE	NODE2VEC	NETSMF	FREDE	LOUVAINNE	RANDNE	NODESKETCH	NODESIG
BlogCat	0.317	0.352	0.376	0.368	0.143	0.337	0.381	**0.408**
Cora	0.708	0.813	0.824	0.825	0.711	0.676	0.825	**0.852**
DBLP	0.632	0.632	0.644	0.661	0.496	0.437	**0.847**	0.843
PPI	0.151	0.161	0.170	0.174	0.054	0.161	0.227	**0.236**
Youtube	0.341	–	0.379	–	0.251	0.341	**0.467**	0.465

Table 2.6: Comparison of models for node classification with 50% training ratio.

	HOPE	NODE2VEC	NETSMF	FREDE	LOUVAINNE	RANDNE	NODESKETCH	NODESIG
BlogCat	0.326	0.345	0.377	0.381	0.165	0.340	0.398	**0.420**
Cora	0.797	0.831	0.831	0.846	0.721	0.693	0.872	**0.879**
DBLP	0.631	0.631	0.647	0.661	0.499	0.438	**0.903**	0.893
PPI	0.146	0.138	0.163	0.157	0.056	0.145	0.243	**0.246**
Youtube	0.343	–	0.376	–	0.256	0.339	**0.476**	0.471

Table 2.7: Comparison of models for node classification with 90% training ratio.

LINK PREDICTION. The set-up of the link prediction task is similar to the one discussed in Sec. 2.4.2. The features of the node pair samples are computed based on the similarities between the embedding vectors depending on the algorithm we use to extract the representations. Table 2.8 gives the Area Under Curve (AUC) scores for NODESIG and the different baseline models. We can observe that NODESIG performs consistently well across the different datasets, even though we learn binary node embeddings. As we will demonstrate in the next paragraph, this helps to achieve a good performance-scalability trade-off.

	HOPE	NODE2VEC	NETSMF	FREDE	LOUVAINNE	RANDNE	NODESKETCH	NODESIG
BlogCat	0.517	0.595	0.691	0.709	0.565	0.608	0.703	**0.822**
Cora	0.665	0.748	0.709	**0.760**	0.684	0.508	0.590	0.737
DBLP	0.769	0.843	0.835	**0.858**	0.789	0.517	0.714	0.856
PPI	0.524	0.616	0.534	0.451	0.570	0.505	0.514	**0.654**
Youtube	0.514	0.533	**0.542**	0.460	0.528	0.502	0.510	0.537

Table 2.8: Comparison of models for the link prediction task.

TIME COMPARISON. We have recorded the elapsed real (wall clock) time of all methods, and the results are provided in Table 2.9. The *Random* network indicates the $\mathcal{G}_{n,p}$ Erdös-Rényi random graph model with 10^5 nodes and $p = 10^{-4}$. Compared to HOPE and NODE2VEC, NODESIG runs significantly faster. Although FREDE is a sketching-based approach, we have observed that the computation of the Personalized PageRank (PPR) matrix requires considerable time. Despite the fact that LOUVAINNE and RANDNE run faster compared to NODESIG, they underperform in both prediction tasks. This experiment further supports the intuition about designing NODESIG as an expressive model that balances accuracy and scalability.

	Blogcat	*Cora*	*DBLP*	*PPI*	*Youtube*	*Random*	**Speedup**
HOPE	97.81	27.32	198.59	32.65	8470.33	1048.52	8.85×
NODE2VEC	1400.44	18.32	161.24	72.16	–	716.07	2.55×
NETSMF	7.78	1.32	10.91	1.90	1624.94	236.30	1.69×
FREDE	1179.79	20.46	2612.84	140.43	–	22386.98	28.33×
LOUVAINNE	0.34	0.06	0.24	0.11	6.86	1.25	0.01×
RANDNE	25.52	3.15	11.55	5.40	449.15	73.11	0.51×
NODESKETCH	64.21	13.42	19.10	14.40	1563.00	101.16	1.59×
NODESIG	17.40	0.74	9.26	2.53	1047.53	38.11	1.00×

Table 2.9: Running time (in seconds) and average speedup.

2.6 DISCUSSION

In this chapter, we introduced four models to learn embeddings in an unsupervised manner. With TNE, we take advantage of the underlying community structure of graphs, leading to the concept of topical node embeddings. In our current study, we have explored three instances of the model relying on different ways to infer the latent communities. In practice, the size of the topic embedding vector learned by (2.4) is a critical factor that affects the performance of TNE. EFGE offers the flexibility to model node co-occurrences with exponential family distributions. We observed that the three different instances of

the model perform similarly. KernelNE and MKernelNE explored a different direction of leveraging random walks in a kernelized matrix factorization framework. Although we did not observe any substantial difference between the single and multiple kernel instances, the kernel models have demonstrated competitive performance overall. We should emphasize the importance of systematically conducting parameter tuning to ensure a fair comparison between random walk embedding models [Gur+22]. Finally, NodeSig's design, following concepts from hashing techniques, allowed to improve scalability without sacrificing effectiveness on downstream tasks. In Chapter 3, we will investigate approaches that aim to extend random walk models to multi-relational and heterogeneous graphs.

3

REPRESENTATION LEARNING ON MULTILAYER AND HETEROGENEOUS GRAPHS

In this chapter, we describe our contributions to representation learning on multilayer and heterogeneous graphs. The chapter begins with an overview of the background material in Sec. 3.1. Then, we introduce BraneExp, a technique that learns node embeddings on multilayer graphs with random walks (Sec. 3.2). We continue by presenting BraneMF, a model that utilizes random walk information in a joint matrix factorization scheme for unsupervised representation learning (Sec. 3.3). For both these models, we consider an evaluation pipeline related to omics data integration tasks in computational biology. The third approach, MDMF, investigates how to infer missing interactions in multilayer heterogeneous graphs, with applications to drug discovery and the task of drug-target interaction prediction (Sec. 3.4). The chapter concludes with a discussion on the three models in Sec. 3.5. The material presented here is based on the following publications:

- Surabhi Jagtap, Abdulkadir Çelikkanat, Aurélie Pirayre, Frédérique Bidard, Laurent Duval, and Fragkiskos D. Malliaros. Multiomics Data Integration for Gene Regulatory Network Inference with Exponential Family Embeddings. In *EUSIPCO*,. 2021.
- Surabhi Jagtap, Abdulkadir Çelikkanat, Aurélie Pirayre, Frédérique Bidard, Laurent Duval, and Fragkiskos D Malliaros. BraneMF: Integration of Biological Networks for Functional Analysis of Proteins. *Bioinformatics* 38:24 (2022), pp. 5383–5389.
- Surabhi Jagtap, Aurélie Pirayre, Frédérique Bidard, Laurent Duval, and Fragkiskos D. Malliaros. BRANEnet: Embedding Multilayer Networks for Omics Data Integration. *BMC Bioinformatics* 23:1 (2022), p. 429.
- Surabhi Jagtap, Aurélie Pirayre, Frédérique Bidard, Laurent Duval, and Fragkiskos D. Malliaros. BRANet: Graph-based Integration of Multi-omics Data with Biological a priori for Regulatory Network Inference. In *CIBB*,. 2021.
- Bin Liu, Dimitrios Papadopoulos, Fragkiskos D. Malliaros, Grigorios Tsoumakas, and Apostolos N Papadopoulos. Multiple similarity drug–target interaction prediction with random walks and matrix factorization. *Briefings in Bioinformatics* 23:5 (2022), bbac353.

3.1 BACKGROUND

Many real-world systems can be described by a set of interacting entities that encompass multiple types of relationships. Such systems can be modeled using the concept of *multilayer graphs*. A multilayer graph, also known as multiplex, multidimensional, or multi-relational graph, is a complex network consisting of multiple interconnected layers or graphs, where each layer represents different types of relationships or interactions between the same or different sets of nodes [Boc+14; Kiv+14; Ber+13]. For instance, in the case of social networks, each layer might correspond to different types of ties

among individuals, while in a protein-protein interaction network, the layers can indicate different ways that proteins or genes are associated with each other (e.g., co-expression or database networks). A schematic representation and an example of a multilayer graph are shown in Fig. 3.1 and 3.2. More formally, a multilayer graph with L layers is defined as a set $\mathcal{G} = \{G^{(l)}\}_{l=1}^{L} = \{(\mathcal{V}_l, \mathcal{E}_l)\}_{l=1}^{L}$ of graphs, where $\mathcal{V}_l = \{v_1, \ldots, v_{n_l}\}$ and $\mathcal{E}_l = \{e_{1_l}, \ldots, e_{m_l}\}$ are the vertex and the undirected edges sets respectively. $n_l = |\mathcal{V}_l|$ and $m_l = |\mathcal{E}_l|$ denote the number of nodes and edges for each layer $l \in \{1, \ldots, L\}$. In many cases, we assume that the layers share the same set of nodes, so $\mathcal{V}_j = \mathcal{V}_l = \mathcal{V}$, and $|\mathcal{V}_j| = |\mathcal{V}_l| = |\mathcal{V}|$ for every $1 \leq j < l \leq L$. We use $\mathbf{A}^{(l)}$ to denote the adjacency matrix of the associated graph layer $G^{(l)}$. In many practical applications, the graph can be *heterogeneous*, where different types of entities interact with each other in various ways. For example, in the case of drug discovery or drug repurposing problems, a graph composed of heterogeneous entities, including drugs, targets (i.e., proteins), and other biomarkers (e.g., diseases, side-effects), can be leveraged.

Our main goal in this chapter is to develop representation learning techniques for multilayer graphs (homogeneous or heterogeneous) capable of effectively integrating information encoded in the different input graph layers. To achieve that, we will combine the ideas presented in Chapter 2 with linear algebra and matrix factorization techniques to learn embeddings in an unsupervised or supervised manner. Other relevant approaches in this direction concern tensor decomposition methods [Sid+17] and multiway data analysis [YA09].

MOTIVATION OF OUR WORK. Before introducing our contributions in the next sections, it is important to briefly discuss the motivation behind developing embedding models for multilayer graphs, which stems from the data integration problem in computational biology [Zit+19; Yue+20]. The cellular system of a living organism is composed of interacting bio-molecules, such as genes, proteins, and metabolites, associated with different biological mechanisms. The increase of omics technologies has favored generating large-scale heterogeneous graph data and the consequent demand for simultaneously using molecular and functional interaction networks: gene co-expression, protein–protein interaction (PPI), genetic interaction, and metabolic networks. However, relying solely on independent analysis of such data is constrained to identifying correlations, primarily representing reactive processes rather than causative ones. On the contrary, these datasets constitute rich sources of information at different molecular levels, and their effective integration is essential to understanding cell functioning and their building blocks (proteins) [Sub+20]. Fusing diverse omics data types is often expected to uncover potential causative changes, such as gene regulation or signal transduction, that underlie specific phenotypes or treatment targets. Therefore, it is necessary to learn informative representations (i.e., embeddings) of bio-molecules that encode interactions across different data sources, which are not fully captured by features extracted directly from individual modalities. This will further enable us to understand biological mechanisms, for instance, gene regulation, protein function prediction, or drug–target identification from a machine learning perspective [Nel+19; Yue+20].

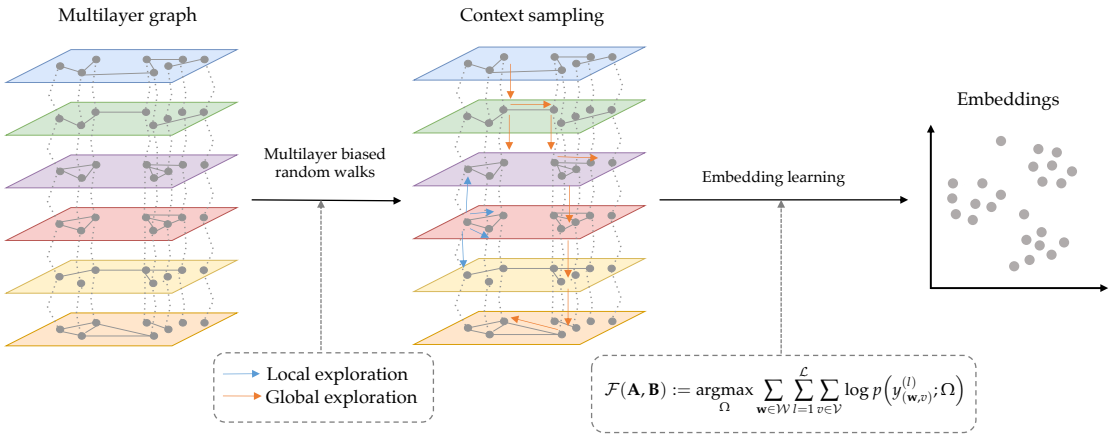


Figure 3.1: Schematic representation of the BRANEEXP model.

3.2 LEARNING EMBEDDINGS ON MULTILAYER GRAPHS WITH RANDOM WALKS

This section is based on material from a conference article co-authored with Surabhi Jagtap, Abdulkadir Çelikkanat, Aurélie Pirayre, Frédérique Bidard, and Laurent Duval published in the *European Signal Processing Conference (EUSIPCO)* [Jag+21b].

In this section, we will introduce our first model, BRANEEXP, which constitutes a simple extension of the EFGE model discussed in Sec. 2.3 to multilayer graphs. The methodology presented here can be applied to any embedding technique that relies on random walk sampling. BRANEEXP has the following conceptual advances: (i) it preserves both the intra-layer and inter-layer interactions, thereby learning rich features; (ii) it is quite an effective method as it uses a properly-chosen conditional probability distribution model to learn low-dimensional node features from all input networks. We first describe how relevant node pairs are sampled with multilayer random walks—a key step towards integrating information across different layers (views) of the graph. Then, we learn node representations by modeling the underlying interactions among nodes with exponential family distributions. A schematic representation of BRANEEXP is given in Fig. 3.1.

CONTEXT SAMPLING WITH MULTILAYER BIASED RANDOM WALKS. In our approach, sampling random walks holds significant importance as it serves two purposes: modeling interactions among nodes and facilitating data integration across input layers. Due to the complex multilayer graph structure, each layer may possess a completely different topological structure, and nodes in the graph layers may have divergent roles. Thus, our task becomes more challenging compared to classical graph representation learning approaches described in Sec. 2. Therefore, we leverage random walks to sample nodes sharing similar characteristics across different graph layers—with the objective being to learn node representations in a lower-dimensional space that capture the underlying patterns shared across these network layers.

To extract nodes' context in a multilayer graph, we leverage a biased random walk sampling procedure that explores local and global structures in the multilayer graph [NM18] (this is a random walk strategy that we have introduced in the past for sampling biased random walks in single-layer graphs). Local exploration helps discover the clustering structure around the node of interest, whereas global exploration contributes to capturing global associations within nodes in the graph. Both exploration types are

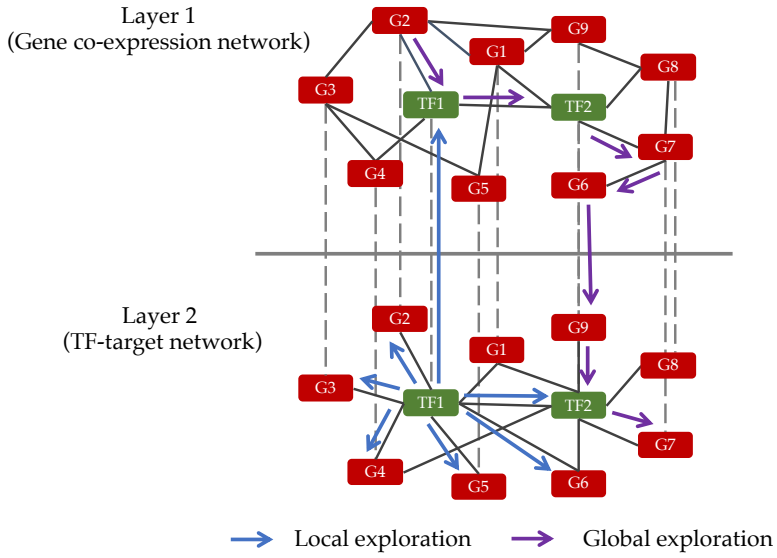


Figure 3.2: Multilayer random walk sampling with BRANEEXP.

particularly important within the biological context of our application domains. Let us consider, for instance, gene co-expression networks with the relationships between regulators, such as transcription factors (TFs) and the target genes they control. In this case, the relationship between a TF and its target gene is a local structure, while functional relationships among the co-expressed genes are more related to global structure in the graph. A schematic representation of the process is shown in Fig. 3.2. The model combines both types of exploration (i.e., local and global) with a decay parameter α to control the importance of nodes with respect to their distance from the node of interest. More formally, for each node $v_i \in \mathcal{V}$, a proximity score τ_i is computed to estimate how far the candidate node v_i is from the source node. When the i-th node in the walk is discovered, the proximity score of every node adjacent to that is increased by α^{i-1} and $\bar{\alpha}^{i-1}$, for nodes in the same and different layer respectively, where α, $\bar{\alpha} \in [0, 1]$. Then, the probability of selecting the next node for the current walk is computed based on the proximity scores of the neighborhood nodes of the most recently visited node. For local explorations, the probability of a node being the next one in the random walk sequence is proportional to its proximity score, i.e., $p_i = \frac{\tau_i}{\sum_{w \in \mathcal{N}(u)} \tau_w}$. In the case of global exploration, the probability is set to be inversely proportional to that score, i.e., $p_i = \frac{1/\tau_i}{\sum_{w \in \mathcal{N}(u)} 1/\tau_w}$, where u is the most recently visited node, and $\mathcal{N}(u)$ defines the set of neighbors of u. Thus, given a desired exploration strategy, the context set for each node v_i can be extracted.

LEARNING EMBEDDINGS WITH EXPONENTIAL FAMILY DISTRIBUTIONS. After sampling random walks in the multilayer graph, the representation learning process follows the principles presented in Sec. 2.3, aiming to maximize the co-occurrence probability of nodes within a certain distance from the node of interest. Specifically, we employ the EFGE-BERN model, in which node co-occurrences within random walk sequences are modeled with a Bernoulli distribution, as given in Eq. (2.10). As we will present shortly in Sec. 3.3.3, the embeddings computed by the BRANEEXP model have the ability to perform well on a variety of downstream prediction tasks.

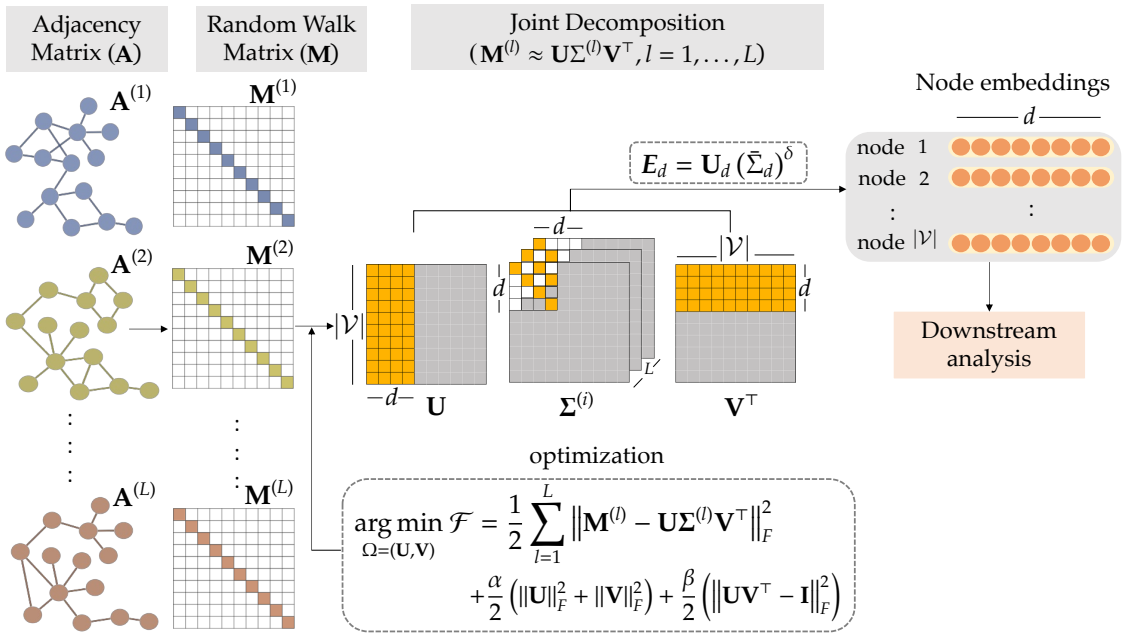


Figure 3.3: Schematic representation of BraneMF.

3.3 MULTILAYER GRAPH EMBEDDINGS WITH JOINT MATRIX FACTORIZATION

This section is based on material from two journal articles co-authored with Surabhi Jagtap, Abdulkadir Çelikkanat, Aurélie Pirayre, Frédérique Bidard, and Laurent Duval published in *Bioinformatics* [Jag+22b] and *BMC Bioinformatics* [Jag+22a]. A preliminary version of this work was published in the *International Conference on Computational Intelligence Methods for Bioinformatics and Biostatistics (CIBB)* [Jag+21a].

One of the limitations of the BraneExp model presented in Sec. 3.2 is that it requires to explicitly sample random walk sequences from a multilayer graph. Besides the computational cost that this step incurs, it is practically difficult to control the importance of each individual layer in the learned node representations. To address this challenge, in this section, we will present BraneMF, an integration framework to learn embeddings from a multilayer graph. A schematic representation of the model is given in Fig. 3.3. BraneMF follows a two-step approach. First, it computes a random walk matrix that effectively captures node proximity between nodes of the same layer. Then, it learns node embeddings through a joint matrix factorizing scheme.

3.3.1 *DeepWalk Random Walk Matrix*

To encode the topological properties of each input graph layer, we leverage a random walk matrix that is directly related to the DeepWalk model [PARS14]. Following the discussion in Sec. 2.1, let $\mathcal{W}$ be the set of sampled random walks. DeepWalk assumes a corpus of node sequences represented as $v_1, v_2, \ldots, v_{\mathcal{L}}$, where $\mathcal{L}$ is the random walk length. Building upon the observations by Levy and Goldberg [LG14] on SkipGram with negative sampling, Qiu *et al.* [Qiu+18b] have shown that DeepWalk when negative

sampling is employed, is equivalent to factorizing the following pointwise mutual information matrix:

$$\underbrace{\log\left(\frac{\#(v,c)|\mathcal{W}|}{\#(v).\#(c)}\right)-\log k}_{\text{SkipGram}} \approx \underbrace{\log\left(\frac{\text{vol}(G)}{\gamma}\left(\sum_{r=1}^{\gamma}\mathbf{P}^r\right)\mathbf{D}^{-1}\right)-\log k}_{\text{DeepWalk matrix } \hat{\mathbf{M}}}.$$

Here we assume that graph $G = (\mathcal{V}, \mathcal{E})$ is undirected, connected, and non-bipartite. On the left-hand side, $\#(v,c), \#(v)$, and $\#(c)$ denote respectively the number of times node-context pair (v,c), node v, and context c appear in $\mathcal{W}$, while k is the number of negative samples. On the right-hand side and the definition of the DeepWalk matrix $\hat{\mathbf{M}}$, $\mathbf{D}$ is the diagonal degree matrix of G, $\mathbf{P}$ is the transition matrix $\mathbf{D}^{-1}\mathbf{A}$, γ is the window size, and $\text{vol}(G) = \sum_{u,v}\mathbf{A}(u,v)$. The logarithm is applied element-wise, which makes $\hat{\mathbf{M}}$ ill-defined. Thus, we consider the shifted positive pointwise mutual information (PPMI) matrix defined as

$$\mathbf{M} := \log\left(\max\left(1, \frac{\text{vol}(G)}{k\gamma}\left(\sum_{r=1}^{\gamma}\mathbf{P}^r\right)\mathbf{D}^{-1}\right)\right). \tag{3.1}$$

Based on this formulation, we obtain the set of DeepWalk matrices $\mathcal{M} = \{\mathbf{M}^{(l)}\}_{l=1}^{L}$ for the different graph layers $G^{(l)}, l = 1, \ldots, L$ of the multilayer graph $\mathcal{G}$. As we will present in the following section, this set of matrices can be jointly decomposed to learn embeddings in a multilayer graph.

3.3.2 *Joint Representation Learning for Multilayer Graphs*

The set of matrices $\mathbf{M}$, as computed by (3.1), captures node proximity that still represents high-dimension node features. Therefore, we want to obtain low-dimension integrated node embeddings that could be easily fed to any downstream machine learning task of interest. Our integration framework is developed on the construction of random walk matrices, on which joint matrix factorization is eventually performed. To learn the spectrum of one layer in graph $\mathcal{G}$, the singular values and singular vectors of its DeepWalk matrix $\mathbf{M}$ can be obtained using SVD as $\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$, where $\mathbf{U}$ and $\mathbf{V}$ correspond to the left and right singular vector matrices, and $\mathbf{\Sigma}$ is the diagonal singular value matrix. In the case of a multilayer graph $\mathcal{G}$ composed of L layers, we have L symmetric DeepWalk matrices. As a natural extension, we propose to approximate each matrix $\mathbf{M}^{(l)}$ by a set of jointly decomposed singular vector and singular value matrices shared by all layers, given by $\mathbf{M}^{(l)} \approx \mathbf{U}\mathbf{\Sigma}^{(l)}\mathbf{V}^\top, l \in \{1, \ldots, L\}$. The correspondence above keeps, where $\mathbf{U}$ and $\mathbf{V}^\top$ are orthogonal matrices containing the joint singular vectors and $\mathbf{\Sigma}^{(l)} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ contains the corresponding singular values in the layer l. The minimization of the following objective function $\mathcal{F}$ yields $\mathbf{U}$ and $\mathbf{V}$:

$$\underset{\Omega=(\mathbf{U},\mathbf{V})}{\text{argmin}}\, \mathcal{F} := \frac{1}{2}\sum_{l=1}^{L}\|\mathbf{M}^{(l)} - \mathbf{U}\mathbf{\Sigma}^{(l)}\mathbf{V}^\top\|_F^2 + \frac{\alpha}{2}(\|\mathbf{U}\|_F^2 + \|\mathbf{V}\|_F^2) + \frac{\beta}{2}\|\mathbf{U}\mathbf{V}^\top - \mathbf{I}\|_F^2, \tag{3.2}$$

where $\mathbf{I} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ is the identity matrix, and $\|\cdot\|_F$ denotes the Frobenius norm. The first term of the objective function $\mathcal{F}$ measures the overall approximation error when all layers are decomposed over $\mathbf{U}$ and $\mathbf{V}^\top$. The second term, the norms of $\mathbf{U}$ and $\mathbf{V}^\top$, is added to improve numerical stability for the solutions. The last term is a constraint to ensure

that $\mathbf{V}^\top$ is close to the inverse of $\mathbf{U}$ ($\mathbf{M}^{(l)}$ is a symmetric matrix, thus its SVD can be given by $\mathbf{U}\mathbf{\Sigma}\mathbf{U}^{-1}$). Since (3.2) is not jointly convex on $\mathbf{U}$ and $\mathbf{V}^\top$, we adopt an alternating scheme to find a local minimum for $\mathcal{F}$. This scheme involves fixing $\mathbf{V}^\top$ initially and optimizing $\mathbf{U}$, and vice versa, as outlined in the work by Dong *et al.* [Don+12]. For parameter initialization in the optimization problem, we suggest to compute the SVD of the mean of all matrices $\mathbf{M}^{(l)}$ and initialize $\mathbf{U}$, $\mathbf{\Sigma}$, and $\mathbf{V}^\top$ with the resulting matrices: $\mathbf{U}$ is the set of joint singular vectors, namely a joint spectrum shared by all layers in $\mathcal{G}$, $\bar{\mathbf{\Sigma}}$ is the joint singular value matrix computed by taking the average of $\mathbf{\Sigma}^{(l)}$ matrices. The final embeddings $\mathbf{E}_d \in \mathbb{R}^{|\mathcal{V}|\times d}$ that integrate information across all input graph layers are obtained by the first d columns of $\mathbf{U}$, $\mathbf{U}_d \in \mathbb{R}^{|\mathcal{V}|\times d}$ scaled by the δ-th power of the singular value magnitudes, $(\bar{\mathbf{\Sigma}}_d)^\delta$, i.e., $\mathbf{E}_d = \mathbf{U}_d(\bar{\mathbf{\Sigma}}_d)^\delta$.

The joint SVD process performed by BraneMF is essentially based on integrating information from multiple graph layers by finding a joint spectrum shared by $\{\mathbf{M}^{(l)}\}_{l=1}^{L}$. Thus, BraneMF is an *intermediate integration* model where integration is performed in the learning process of embedding computation [Zit+19]. We have also introduced a more straightforward methodology, referred to as BraneNet [Jag+22a], corresponding to an *early integration* strategy. This model performs the decomposition of a large matrix computed by stacking the DeepWalk matrices of each input layer (similar to the notion of the supra-adjacency matrix in multilayer graphs [Boc+14]).

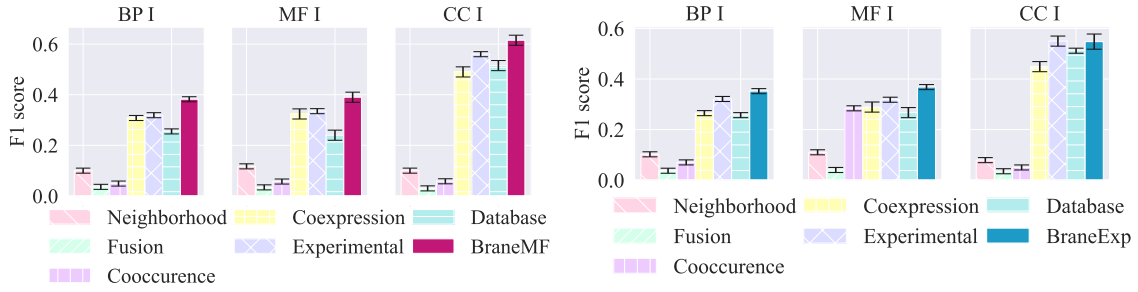
3.3.3 *Experimental Evaluation of BraneExp and BraneMF*

This section provides an overview of the experimental evaluation of BraneExp and BraneMF. For a comprehensive list of experiments, please refer to the corresponding articles [Jag+21b; Jag+22b].

DATASETS. The experiments described here have been conducted in collaboration with IFP Energies nouvelles (https://www.ifpenergiesnouvelles.fr/), an institute that specializes in industrial biotechnology in the context of biofuel production. To substantiate our methodology, we consider a multilayer protein-protein interaction (PPI) network of *Saccharomyces cerevisiae*, a well-studied *yeast* model organism. For this study, we consider six PPI networks for *yeast* extracted from the STRING database [Szk+20]: *Neighborhood*, *Fusion*, *Co-occurrence*, *Co-expression*, *Experimental*, and *Database*. The PPI networks are built for 6,691 proteins. Each of these six networks corresponds to an input graph layer. We have investigated the usefulness of the learned embeddings on various omics inference tasks, including clustering and Gene Ontology (GO) enrichment analysis, protein function prediction, PPI prediction, PPI network reconstruction, and Gene Regulatory Network (GRN) inference. In the next paragraphs, we elaborate in more detail on two of those biological downstream tasks.

BASELINE MODELS. We have considered eight multilayer graph embedding and integration methods to benchmark BraneExp and BraneMF: SNF [Wan+14], Mashup [CBP16], deepNF [GBB18], MultiNet [BK18], Multi-Node2Vec [Wil+21], OhmNet [ZL17], MultiVERSE [PL+21], and Graph2GO [FGZ20].

PROTEIN FUNCTION PREDICTION AS A NODE CLASSIFICATION TASK. The multilayer graph embedding models developed here allow us to obtain low-dimensional protein features, combining different PPI networks. Here, we investigate the ability of these features to predict protein functions, casting the problem to a multi-label node



(a) BRANEMF (b) BRANEEXP

Figure 3.4: Performance of BRANEMF and BRANEEXP applied on individual (single-layer) vs. multilayer networks. Integrating multiple networks outperforms individual network embeddings in protein function prediction.

classification task. For functional annotations of proteins (i.e., node labels), we use Gene Ontology (GO) terms covering three domains: biological process (BP), molecular function (MF), and cellular component (CC) [Con04]. Each category of Gene Ontology is depicted across three levels, namely, levels I, II, and III. Lower levels (level I) represent more specific terms, whereas a higher level (level III) signifies more general terms. We use the learned features to predict functional annotations using an SVM classifier [CL11; Ped+11]. We adopt a 5-fold cross-validation (CV) process to evaluate the classification performance. We split all the annotated proteins into a training set, comprising 80% of the proteins, and a test set, comprising the remaining 20%. All performance results are averaged over ten different CV trials.

First, we investigate the added value of integration for protein function prediction. To do this, we have learned embeddings of proteins for each input network layer and subsequently performed classification. We then compare the performance of the features learned from individual input networks with those obtained from the integrated ones. The results for BP, MF, and CC (level I) are shown in Fig. 3.4. Our observations indicate that integration yields better performance for both BRANEMF and BRANEEXP than individual network embeddings in the context of protein function prediction. Moreover, the *Experimental*, *Co-expression*, and *Database* networks demonstrate better performance, indicating the significance of these network layers to the prediction task.

Additionally, we have explored the performance of three different network integration strategies: early, intermediate, and late [Zit+19]. In early integration, all input graphs are merged into a single one before the embedding learning process occurs. On the contrary, late integration is performed after the model is applied to each network by concatenating the learned embeddings. BRANEMF is an intermediate integration model where integration is performed in the embedding learning process. To demonstrate the effectiveness of intermediate integration, we have compared BRANEMF to BRANEMF-early and BRANEMF-late. In BRANEMF-early, the DEEPWALK matrix $\mathbf{M}$ is computed from the adjacency matrix of the graph obtained by taking the union of all six input graph layers. In BRANEMF-late, the node embeddings are learned independently for each layer, which are then averaged to obtain the final protein features. As we can observe from the results given in Fig. 3.5, BRANEMF outperforms the rest of the integration strategies for all three types of functional annotations. In particular, there is an increase of 2% in the accuracy of BP I compared to BRANEMF-early and an increase of 10% compared to the BRANEMF-late integration model. This improvement can be partially attributed to the fact that separately computing the random walk matrices of each individual layer uncovers

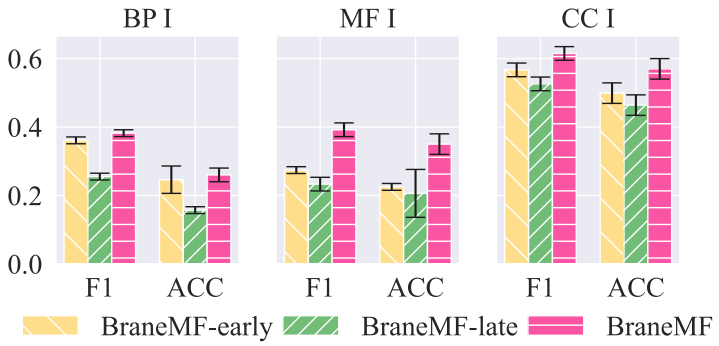


Figure 3.5: Performance of BraneMF compared to early and late integration strategies (F_1 score and accuracy).

Method	BP I	MF I	CC I
SNF	$0.199_{\pm 0.01}$	$0.104_{\pm 0.00}$	$0.206_{\pm 0.01}$
Mashup	$0.277_{\pm 0.00}$	$0.263_{\pm 0.02}$	$0.520_{\pm 0.02}$
deepNF	$0.341_{\pm 0.01}$	$0.342_{\pm 0.02}$	$0.564_{\pm 0.02}$
MultiNet	$0.335_{\pm 0.01}$	$0.353_{\pm 0.02}$	$0.532_{\pm 0.02}$
Multi-Node2Vec	$0.331_{\pm 0.01}$	$0.323_{\pm 0.01}$	$0.511_{\pm 0.01}$
OhmNet	$0.321_{\pm 0.01}$	$0.300_{\pm 0.01}$	$0.512_{\pm 0.01}$
MultiVERSE	$0.312_{\pm 0.01}$	$0.294_{\pm 0.01}$	$0.502_{\pm 0.02}$
Graph2GO	$0.340_{\pm 0.01}$	$0.355_{\pm 0.01}$	$\underline{0.564}_{\pm 0.02}$
BraneExp	$\underline{0.352}_{\pm 0.01}$	$\underline{0.368}_{\pm 0.01}$	$0.548_{\pm 0.03}$
BraneMF	$\mathbf{0.382}_{\pm 0.01}$	$\mathbf{0.392}_{\pm 0.02}$	$\mathbf{0.615}_{\pm 0.02}$

Table 3.1: Comparison of models on the protein function prediction task using the F_1 score.

compressed topological patterns that are difficult to identify in the combined network (BraneMF-early model), where different edge types are not distinguished.

Finally, in Table 3.1, we compare the performance of BraneMF and BraneExp to the baseline methods introduced earlier. We observe that protein function prediction based on BraneMF substantially outperforms other integration methods in assigning a previously unseen protein to its known functional categories. This performance gain is consistent across all three Gene Ontology terms.

PROTEIN-PROTEIN INTERACTION PREDICTION. The interactome is the map of PPIs that can occur in an organism. Nevertheless, it is still an open question whether experimental techniques will ever discover the complete interactome of any organism. In this context, predictive methods have gained popularity in computational biology for uncovering the wiring patterns of proteins. The effective integration of PPIs from different data sources (experimental and/or computational) can further help to infer a nearly complete set of the interactome. In this task, we aim to predict missing (unseen) protein-protein interactions, casting the problem to a link prediction task. Specifically, we use PPIs from the 2015 and 2021 STRING networks to form training and test sets. We form the positive training set from PPIs that did not change from 2015 to 2021 and the positive test set from the PPIs that did not exist in 2015 but were discovered in 2021. The learned embeddings of protein u and v, given by $\mathbf{E}_d[u]$ and $\mathbf{E}_d[v]$, are converted into edge feature vectors by applying the coordinate-wise Hadamard product or cosine similarity operations. The results are shown in Table 3.2. We have observed that BraneMF exhibits competitive

Method	AUPR-H	AUROC-H	AUPR-C	AUROC-C
SNF	0.637	0.628	0.575	0.559
MASHUP	0.757	0.743	0.712	**0.707**
DEEPNF	0.764	0.747	0.490	0.480
MULTINET	0.735	0.724	0.490	0.480
MULTI-NODE2VEC	0.526	0.528	0.511	0.509
OHMNET	0.513	0.514	0.516	0.516
MULTIVERSE	0.500	0.501	0.501	0.501
GRAPH2GO	0.721	0.757	0.502	0.498
BRANEEXP	0.777	**0.760**	0.683	0.680
BRANEMF	**0.783**	0.747	**0.725**	0.682

Hadamard product (-H); cosine similarity (-C).

Table 3.2: Comparison of models on the PPI prediction task using AUROC and AUPR scores.

and consistent behavior across almost all evaluation metrics for PPI prediction, achieving 1.5% higher performance (AUPR-H) than BRANEEXP, the second-best performing model. DEEPNF and MASHUP also perform well under specific evaluation metrics.

3.4 LINK PREDICTION ON MULTILAYER HETEROGENEOUS GRAPHS

This section is based on material from a journal article co-authored with Bin Liu, Dimitrios Papadopoulos, Grigorios Tsoumakas, and Apostolos N. Papadopoulos published in *Briefings in Bioinformatics* [Liu+22a].

This section considers a link prediction task on multi-relational heterogeneous graphs. Contrary to the models discussed in Sec. 3.2 and Sec. 3.3 that aim to learn embeddings in an unsupervised manner, here we combine embedding generation and interaction prediction via a matrix factorization framework. The practical application concerns the problem of drug-target interaction (DTI) prediction in drug discovery [Bag+21]. Although *in vitro* experimental testing can verify DTIs, it suffers from extremely high time and monetary costs. Using computational methods (*in silico*) to accurately identify reliable interactions between drugs and proteins, often by leveraging heterogeneous information from diverse data sources, can significantly enhance the development of effective pharmaceuticals.

In the past, the chemical structure of drugs and the protein sequence of targets were the main sources of information for inferring candidate DTIs [Liu+16b; PVT21]. Recently, with the advancements in clinical medical technology, abundant drug and target-related biological data from multifaceted sources have been exploited to boost the accuracy of DTI prediction. Matrix factorization (MF) and kernel-based methods have utilized multiple types of drug and target similarities derived from heterogeneous information by integrating them into a single drug and target similarity [Zhe+13; DTG20; OAB18]. However, in doing so, they often discard the distinctive information possessed by each similarity view. As an alternative, graph-based approaches consider the diverse drug and target data as a heterogeneous DTI network that describes multiple aspects of drug and target relations. Then, topology-preserving embeddings of drugs and targets are learned either via neural network models [Wan+19] or with random walks [Luo+17; AY21b]. Although these methods can model high-order node proximity efficiently, they typically perform embedding generation and interaction prediction as two independent tasks.

Hence, their embeddings are learned in an unsupervised manner, failing to preserve the topology information from the interaction network.

In this section, we will introduce the *Multiple Similarity DeepWalk Matrix Factorization* (MDMF) model, which incorporates multilayer heterogeneous DTI network embedding generation and DTI prediction within a unified optimization framework. The model learns vector embeddings of drugs and targets that not only capture the multilayer network topology via factorizing a DeepWalk matrix with information from diverse data sources but also preserve the layer-specific local invariance with the graph Laplacian for each drug and target similarity view. Based on this formulation, we instantiate two models that leverage surrogate losses to optimize two essential evaluation measures in DTI prediction, namely AUPR and AUC.

3.4.1 *Problem Formulation*

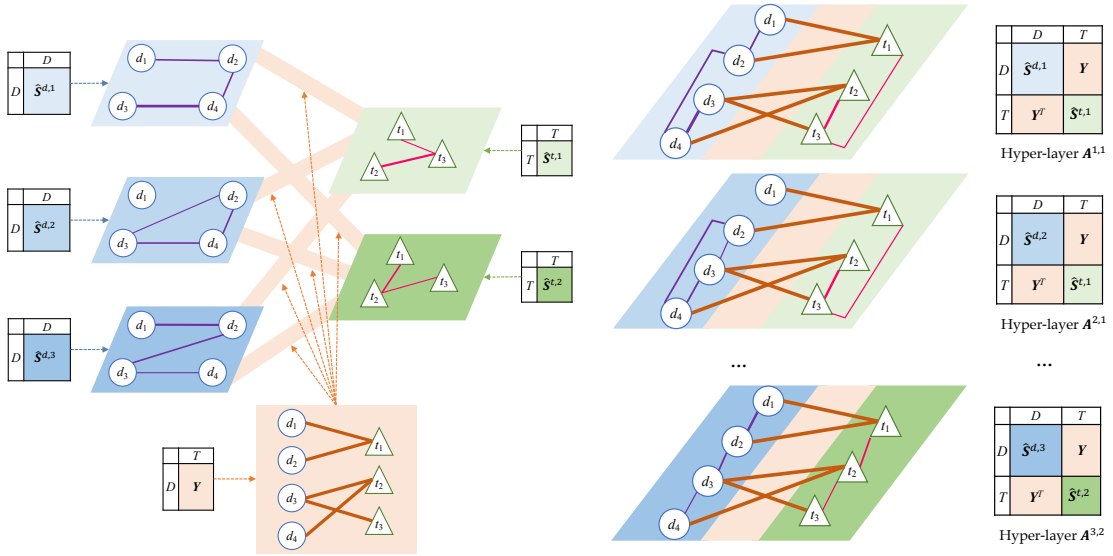
Given a drug set $D = \{d_i\}_{i=1}^{n_d}$ and a target set $T = \{t_i\}_{i=1}^{n_t}$, the relation between drugs (targets) can be assessed in various ways, which are represented by a set of similarity matrices $\{\mathbf{S}^{d,h}\}_{h=1}^{m_d}$ ($\{\mathbf{S}^{t,h}\}_{h=1}^{m_t}$), where $\mathbf{S}^{d,h} \in \mathbb{R}^{n_d \times n_d}$ ($\mathbf{S}^{t,h} \in \mathbb{R}^{n_t \times n_t}$) and m_d (m_t) is the number of relation types for drugs (targets). In addition, let the binary matrix $\mathbf{Y} \in \{0,1\}^{n_d \times n_t}$ indicate the interactions between drugs in D and targets in T, where $Y_{ij} = 1$ denotes that d_i and t_j interact with each other, and $Y_{ij} = 0$ otherwise. A DTI dataset for D and T consists of $\{\mathbf{S}^{d,h}\}_{h=1}^{m_d}$, $\{\mathbf{S}^{t,h}\}_{h=1}^{m_t}$ and $\mathbf{Y}$. Let (d_x,t_z) be a test drug-target pair, $\{\bar{\mathbf{s}}_x^{d,h}\}_{h=1}^{m_d}$ be a set of n_d-dimensional vectors storing the similarities between d_x and D, and $\{\bar{\mathbf{s}}_z^{t,h}\}_{h=1}^{m_d}$ be a set of n_t-dimensional vectors storing the similarities between t_z and T. A DTI prediction model predicts a real-valued score $\hat{Y}_{xz}$ indicating the confidence of the affinity between d_x and t_z. In addition, $d_x \notin D$ ($t_z \notin T$), which does not belong to the training set, is considered as the new drug (target). There are four prediction settings according to whether the drug and target involved in the test pair are training entities: (S1): predict the interaction between $d_x \in D$ and $t_z \in T$; (S2): predict the interaction between $d_x \notin D$ and $t_z \in T$; (S3): predict the interaction between $d_x \in D$ and $t_z \notin T$; (S4): predict the interaction between $d_x \notin D$ and $t_z \notin T$.

MATRIX FACTORIZATION FOR DTI PREDICTION. In DTI prediction, MF methods typically learn two vectorized representations of drugs and targets that approximate the interaction matrix $\mathbf{Y}$ by minimizing the following objective:

$$\min_{\mathbf{U},\mathbf{V}} \mathcal{F}(\hat{\mathbf{Y}}, \mathbf{Y}) + \mathcal{R}(\mathbf{U}, \mathbf{V}), \tag{3.3}$$

where $\hat{\mathbf{Y}} = f(\mathbf{U}\mathbf{V}^\top) \in \mathbb{R}^{n_d \times n_t}$ is the predicted interaction matrix, f is either the identity function ω for standard MF [Ezz+17] or the element-wise logistic function σ for Logistic MF [Liu+16b], and $\mathbf{U} \in \mathbb{R}^{n_d \times r}$, $\mathbf{V} \in \mathbb{R}^{n_t \times r}$ are r-dimensional drug and target latent features (embeddings), respectively. The objective in (3.3) includes two parts: $\mathcal{F}(\hat{\mathbf{Y}}, \mathbf{Y})$ is the loss function to evaluate the inconsistency between the predicted and ground truth interaction matrix, and $\mathcal{R}(\mathbf{U}, \mathbf{V})$ concerns the regularization of the learned embeddings. Given a test drug-target pair (d_x, t_z), its prediction with a specific instantiation of f is computed based on the embeddings of d_x ($\mathbf{U}_x \in \mathbb{R}^r$) and t_z ($\mathbf{V}_z \in \mathbb{R}^r$):

$$\hat{Y}_{xz} = \begin{cases} \mathbf{U}_x \mathbf{V}_z^\top, \text{if } f = \omega \\ \left(1 + \exp(-\mathbf{U}_x \mathbf{V}_z^\top)\right)^{-1}, \text{if } f = \sigma. \end{cases} \tag{3.4}$$



(a) A multilayer heterogeneous DTI network

(b) The DTI network on the left reformulated as multiple hyper-layers

Figure 3.6: Representing a DTI dataset with three drug and two target similarities as a network. (a) A multilayer heterogeneous network including three drug layers (blue), two target layers (green), and six identical bipartite interaction subnetworks (orange) connecting drug and target nodes in each layer. (b) Six multiple hyper-layers, where each of them is composed of a drug and a target layer along with the interaction subnetwork.

3.4.2 *Multiple Similarity Matrix Factorization with DeepWalk Regularization*

A DTI dataset, associated with multiple drug and target similarities, can be viewed as a multiplex heterogeneous network G^{DTI}. This can be done by treating drugs and targets as two types of nodes and by considering non-zero similarities and interactions as edges connecting two homogeneous and heterogeneous entities, respectively, where the weight of each edge equals the corresponding similarity or interaction value. In a DTI dataset, the interaction matrix $\mathbf{Y}$ is typically more sparse than the similarity matrices, causing the similarity-derived edges linking two drugs (targets) to be markedly outnumber compared to the critical bipartite interaction edges. To balance the distribution of different types of edges and stress relations of more similar entities in the DTI network, we replace each original dense similarity matrix $\mathbf{S}^{d,h}$ with the sparse adjacency matrix $\hat{\mathbf{S}}^{d,h}$ of its corresponding k-nearest neighbors (k-NNs) graph. Formally, G^{DTI} consists of three parts: (i) $G^d = \{\hat{\mathbf{S}}^{d,h}\}_{h=1}^{m_d}$, which is a multilayer drug subnetwork containing m_d layers, with $\hat{\mathbf{S}}^{d,h}$ being the adjacency matrix of the h-th drug layer; (ii) $G^t = \{\hat{\mathbf{S}}^{t,h}\}_{h=1}^{m_t}$, which is a multilayer target subnetwork including m_t layers with $\hat{\mathbf{S}}^{t,h}$ denoting the adjacency matrix of the h-th target layer; (iii) $G^Y = \mathbf{Y}$, which is a bipartite interaction subnetwork connecting drug and target nodes in each layer. Figure 3.6a depicts an example DTI network.

To model the interactions in the multilayer heterogeneous DTI network, we leverage the DeepWalk matrix introduced in Sec. 3.3 in Eq. (3.1). However, the DeepWalk matrix cannot be directly calculated for the complex DTI network that includes two multilayer and a bipartite subnetwork. To facilitate its computation, we consider each combination of a drug and a target layer along with the interaction subnetwork as a hyper-layer, and reformulate the DTI network as a multilayer network containing $m_d \cdot m_t$ hyper-layers. The hyper-layer incorporating the i-th drug layer and j-th target layer, is defined by

the adjacency matrix $\mathbf{A}^{i,j} = \begin{bmatrix} \hat{\mathbf{S}}^{d,i} & \mathbf{Y} \\ \mathbf{Y}^\top & \hat{\mathbf{S}}^{t,j} \end{bmatrix}$, upon which G^{DTI} could be expressed as a set of hyper-layers $\{\mathbf{A}^{i,j}\}_{i=1,j=1}^{m_d,m_t}$. Figure 3.6b illustrates the multiple hyper-layer network corresponding to the original DTI network in Fig. 3.6a. Based on this formulation, we compute a DeepWalk matrix $\mathbf{M}^{i,j} \in \mathbb{R}^{(n_d+n_t)\times(n_d+n_t)}$ for each $\mathbf{A}^{i,j}$. Finally, in order to mine multiple DeepWalk matrices effectively, we define a unified DeepWalk matrix for the whole DTI network by aggregating every $\mathbf{M}^{i,j}$:

$$\bar{\mathbf{M}} = \sum_{i=1}^{m_d}\sum_{j=1}^{m_t} w_i^d w_j^t \mathbf{M}^{i,j}, \tag{3.5}$$

where w_i^d and w_j^t are weights of i-th drug and j-th target layers respectively with $\sum_{i=1}^{m_d} w_i^d = 1$ and $\sum_{j=1}^{m_t} w_j^t = 1$. The importance of each hyper-layer in (3.5) is determined by multiplying the weights of its involved drug and target layers.

Let $\mathbf{Q} = [\mathbf{U}\ \mathbf{V}]^\top$ be the concatenation of drug and target embeddings. We encourage $\mathbf{Q}\mathbf{Q}^\top$ to approximate $\bar{\mathbf{M}}$, which enables the learned embeddings to capture the topology information characterized by the holistic DeepWalk matrix. Hence, we derive the DeepWalk regularization term that diminishes the discrepancy between $\bar{\mathbf{M}}$ and $\mathbf{Q}\mathbf{Q}^\top$:

$$\mathcal{R}_{dw}(\mathbf{U},\mathbf{V}) = ||\bar{\mathbf{M}} - \mathbf{Q}\mathbf{Q}^\top||_F^2. \tag{3.6}$$

Considering the fact that $\mathbf{A}^{i,j}$ includes four blocks, $\mathcal{R}_{dw}(\mathbf{U},\mathbf{V})$ can be expressed as

$$\mathcal{R}_{dw}(\mathbf{U},\mathbf{V}) = ||\bar{\mathbf{M}}_{S_d} - \mathbf{U}\mathbf{U}^\top||_F^2 + 2||\bar{\mathbf{M}}_Y - \mathbf{U}\mathbf{V}^\top||_F^2 + ||\bar{\mathbf{M}}_{S_t} - \mathbf{V}\mathbf{V}^\top||_F^2. \tag{3.7}$$

However, aggregating all per-layer DeepWalk matrices to the holistic one inevitably leads to a substantial loss of layer-specific topology information. To address this limitation, we employ graph regularization for each sparsified drug (target) layer to preserve per-layer drug (target) proximity in the embedding space, i.e., similar drugs (targets) in each layer are likely to have similar latent features. By replacing $\mathcal{R}(\mathbf{U},\mathbf{V})$ in (3.3) with the above regularization terms, we arrive at the objective of the MDMF model:

$$\begin{aligned} \min_{\mathbf{U},\mathbf{V}} \quad & \mathcal{F}(\hat{\mathbf{Y}},\mathbf{Y}) + \frac{\lambda_M}{2}\mathcal{R}_{dw}(\mathbf{U},\mathbf{V}) + \frac{\lambda_d}{2}\sum_{i=1}^{m_d} w_i^d \mathrm{tr}(\mathbf{U}^\top \mathbf{L}_i^d \mathbf{U}) \\ & + \frac{\lambda_t}{2}\sum_{j=1}^{m_t} w_j^t \mathrm{tr}(\mathbf{V}^\top \mathbf{L}_j^t \mathbf{V}) + \frac{\lambda_r}{2}\left(||\mathbf{U}||_F^2 + ||\mathbf{V}||_F^2\right), \end{aligned} \tag{3.8}$$

where $\mathbf{L}_i^d = \mathrm{diag}(\hat{\mathbf{S}}^{d,i}\mathbf{e}) - \hat{\mathbf{S}}^{d,i}$ and $\mathbf{L}_j^t = \mathrm{diag}(\hat{\mathbf{S}}^{t,j}\mathbf{e}) - \hat{\mathbf{S}}^{t,j}$ are graph Laplacian matrices of $\hat{\mathbf{S}}^{d,i}$ and $\hat{\mathbf{S}}^{t,j}$ respectively, λ_M, λ_d, λ_t, and λ_r are regularization coefficients.

OPTIMIZING THE AREA UNDER THE CURVE WITH MDMF. AUPR and AUC are two widely used *area under the curve* metrics in DTI prediction. Modeling differentiable surrogate losses that optimize these two metrics can lead to prediction performance improvement [LT21]. Therefore, we instantiate the loss function $\mathcal{F}(\hat{\mathbf{Y}},\mathbf{Y})$ in (3.8) with AUPR and AUC losses, and derive two new models, namely MDMF-AUPR and MDMF-AUC, that optimize the AUPR and AUC metrics, respectively. The exact formulation can be found in our article [Liu+22a]. Finally, we propose an ensemble model, called DWFM2A, which integrates the two surrogate loss MF models by aggregating their predicted scores. Given a test pair (d_x, t_z) along with its predicted scores $\hat{Y}_{xz}^{AUPR}$ and

Setting	Dataset	WkNNIR	MSCMF	NRMFL	GRGMF	MF2A	DRLSM	DTINet	NEDTP	MDMF2A
S1	*NR*	-	0.628	0.640	0.658	0.673	0.642	0.508	0.546	**0.675**
	GPCR	-	0.844	0.860	0.844	0.870	0.835	0.597	0.798	**0.874**
	IC	-	0.936	0.934	0.913	0.943	0.914	0.693	0.906	**0.946**
	E	-	0.818	0.843	0.832	0.858	0.811	0.305	0.78	**0.859**
	Luo	-	0.599	0.603	0.636	0.653	0.598	0.216	0.080	**0.679**
S2	*NR*	0.562	0.531	0.547	0.564	0.578	0.570	0.339	0.486	**0.602**
	GPCR	0.540	0.472	0.508	0.542	0.551	0.532	0.449	0.451	**0.561**
	IC	0.491	0.379	0.479	0.493	0.495	0.466	0.365	0.407	**0.502**
	E	0.405	0.288	0.389	0.415	0.422	0.376	0.173	0.33	**0.428**
	Luo	0.485	0.371	0.458	0.462	0.472	**0.502**	0.187	0.077	0.480
S3	*NR*	0.560	0.505	0.519	0.545	**0.588**	0.546	0.431	0.375	0.582
	GPCR	0.774	0.690	0.729	0.755	0.787	0.757	0.546	0.684	**0.788**
	IC	0.861	0.827	0.838	0.851	0.863	0.855	0.599	0.800	**0.865**
	E	0.728	0.623	0.711	0.715	0.731	0.712	0.313	0.615	**0.738**
	Luo	0.243	0.080	0.204	0.234	0.292	0.248	0.061	0.046	**0.299**
S4	*NR*	**0.309**	0.273	0.278	0.308	0.289	0.236	0.249	0.160	0.286
	GPCR	0.393	0.323	0.331	0.368	**0.407**	0.077	0.306	0.290	**0.407**
	IC	0.339	0.194	0.327	0.347	0.352	0.086	0.264	0.251	**0.356**
	E	0.228	0.074	0.221	0.224	0.235	0.063	0.100	0.112	**0.239**
	Luo	0.132	0.018	0.096	0.106	0.175	0.061	0.035	0.030	**0.182**

Table 3.3: AUPR results in all prediction settings.

$\hat{Y}_{xz}^{AUC}$ obtained from MDMF-AUPR and MDMF-AUC respectively, the final prediction output by MDMF2A is defined as:

$$\hat{Y}_{xz} = \beta \hat{Y}_{xz}^{AUPR} + (1-\beta)\sigma(\hat{Y}_{xz}^{AUC}), \tag{3.9}$$

where $\beta \in [0,1]$ is the trade-off coefficient for the two MF base models, and σ converts $\hat{Y}_{xz}^{AUC}$ and $\hat{Y}_{xz}^{AUPR}$ into the same scale, i.e., $(0,1)$.

3.4.3 *Experimental Evaluation of MDMF*

DATASETS. In our study, we have used five DTI datasets. Four of them are datasets constructed by Yamanishi *et al.* [Yam+08], each one corresponding to a target protein family, namely Nuclear Receptors (*NR*), Ion Channel (*IC*), G-protein coupled receptors (*GPCR*), and Enzyme (*E*). Since the datasets were introduced in 2008, we have further enhanced them with newly discovered interactions between drugs and targets from relevant databases. Four types of drug and target similarities have been used to construct the different layers of the multilayer graph. The last dataset has been provided in the study by Luo *et al.* [Luo+17] (denoted as *Luo*) and contains four drug and three target types of similarity.

BASELINE MODELS. In the sample experiments that we report below, we have compared MDMF2A to seven DTI prediction models, namely WkNNIR [Liu+22b], NRLMF [Liu+16b], MSCMF [Zhe+13], GRGMF [Zha+20], MF2A [LT21], DRLSM [DTG20], DTINet [Luo+17], and NEDTP [AY21b]).

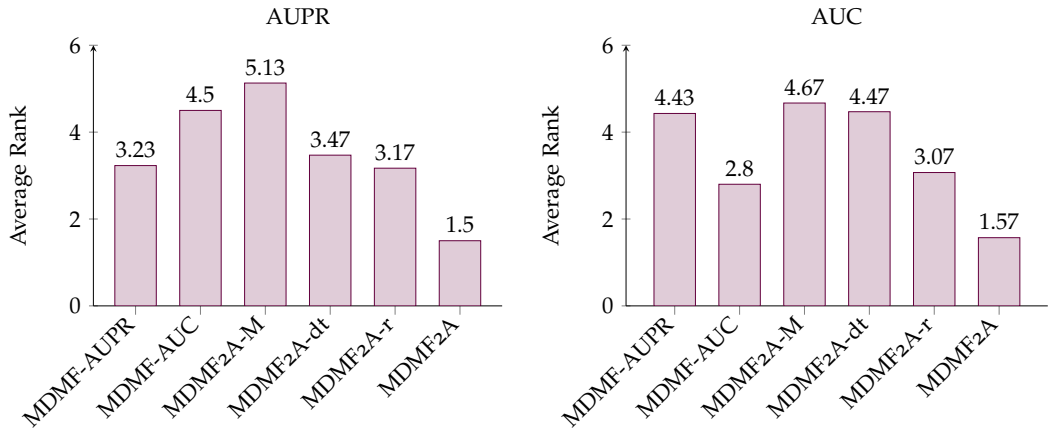


Figure 3.7: The average rank of MDMF2A and its variants over all five datasets and four settings. A lower rank indicates better performance.

DTI PREDICTION RESULTS. Table 3.3 gives the performance of MDMF2A and the baseline models under four prediction settings discussed in Sec. 3.4.1. Here, we report results with the AUPR metric; a similar performance has been observed for AUC. MDMF2A consistently ranks among the best-performing models for both metrics (AUPR and AUC) in all prediction settings (S1 - S4). This demonstrates the effectiveness of our model to sufficiently exploit the topological information embedded in the multilayer heterogeneous DTI network and optimize the two area under the curve metrics. MF2A is the runner-up model. Its inferiority to MDMF2A lies in its failure to consider higher-order proximity captured by random walks and the view-specific information loss caused by aggregating multi-type similarities.

ABLATION STUDY. To demonstrate the effectiveness of the ensemble framework of MDMF2A of (3.9) and the regularization terms employed in (3.8), we conduct an ablation study by considering five degenerate variants of the model: (i) MDMF-AUPR by setting β to 1; (ii) MDMF-AUC by setting β to 0; (iii) MDMF2A-M excludes the DEEPWALK matrix regularization, i.e., $\lambda_M = 0$; (iv) MDMF2A-dt does not consider layer-specific graph regularization i.e., $\lambda_d = \lambda_t = 0$; (v) MDMF2A-r ignores Tikhonov regularization, i.e., $\lambda_r = 0$. The average rank of MDMF2A and its five variants over all five datasets and four settings are shown in Fig. 3.7. Firstly, it is important to note that while MDMF-AUPR and MDMF-AUC demonstrate good performance in the specific metric they optimize, their performance deteriorates in the other metric. MDMF2A outperforms its two base models in terms of both AUC and AUPR, providing strong evidence of the successful integration of two single metric-aware models, resulting in performance improvement. Concerning the regularization terms, we observe that removing any term would trigger severe performance degradation, confirming their importance in MDMF-based models. More specifically, the DEEPWALK matrix regularization term, which captures higher-order proximity in the graph topology, is the most crucial one. Layer-specific graph regularization that preserves distinctive information from each view comes next and contributes more to AUC. Although the damage caused by neglecting Tikhonov regularization is less than the other two terms, its function to avoid overfitting cannot be ignored.

3.5 DISCUSSION

This chapter examined representation learning techniques for multilayer and heterogeneous graphs. Our first approach, BRANEEXP, extended random walk models to multilayer graphs. Despite the good performance, the model requires sampling random walk sequences, which renders training and parameter tuning challenging. Our second model, BRANEMF, leveraged the DEEPWALK matrix within a joint matrix factorization framework to learn embeddings on a multilayer graph. This intermediate integration model achieved highly competitive performance in various protein analysis tasks. Finally, we studied the problem of DTI prediction by effectively mining the graph structure of a multilayer heterogeneous network involving diverse drug and target similarities. MDMF relied on a matrix factorization framework with DEEPWALK regularization, allowing also to simultaneously optimize the AUPR and AUC surrogate loss functions.

4

REPRESENTATION LEARNING WITH GRAPH NEURAL NETWORKS

GRAPH NEURAL NETWORK (GNN) models have emerged as the current "workhorse" in graph representation learning and geometric deep learning, in general. This chapter presents an overview of our contributions to this field. After Sec. 4.1, which discusses the basic background concepts, we present three methodologies that aim to address challenging problems while learning and making predictions with GNNs. First, we introduce the SJRL model that allows the design of deep GNNs alleviated from over-smoothing and over-squashing (Sec. 4.2). Then, we present HOSCPOOL, a hierarchical clustering and pooling model for GNNs (Sec. 4.3). The third approach, GRAPHSVX, deals with the design of explainability models for GNNs (Sec. 4.4). The content of this chapter is based on the following publications:

- Jhony H. Giraldo, Konstantinos Skianis, Thierry Bouwmans, and Fragkiskos D. Malliaros. On the Trade-off between Over-Smoothing and Over-Squashing in Deep Graph Neural Networks. In *CIKM,*. 2023.
- Alexandre Duval and Fragkiskos D. Malliaros. Higher-order Clustering and Pooling for Graph Neural Networks. In *CIKM,*. 2022, pp. 426–435.
- Alexandre Duval and Fragkiskos D. Malliaros. GraphSVX: Shapley Value Explanations for Graph Neural Networks. In *ECML PKDD,*. 2021, pp. 302–318.

4.1 BACKGROUND

In this chapter, we consider undirected, connected, and unweighted graphs. $\mathbf{A} \in \{0,1\}^{N\times N}$ is the adjacency matrix of G such that $\mathbf{A}(i,j) = 1$ if $(i,j) \in \mathcal{E}$ and $\mathbf{A}(i,j) = 0$ otherwise. Moreover, $\mathbf{D} \in \mathbb{R}^{N\times N}$ is the diagonal degree matrix of G such that $\mathbf{D}(i,i) = \sum_{j=1}^{N} \mathbf{A}(i,j)\ \forall\ i = 1,\dots,N$, and $d_i = \mathbf{D}(i,i)$. $\mathbf{L} = \mathbf{D} - \mathbf{A}$ is the positive semi-definite combinatorial Laplacian operator. Similarly, $\mathbf{L}_{\text{sym}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ is the symmetrically normalized Laplacian matrix with eigenvalues $0 = \lambda_1 < \lambda_2 \leq \cdots \leq \lambda_N \leq 2$ and corresponding eigenvectors $\{\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_N\}$. A detailed discussion of concepts in spectral graph theory can be found at [Chu97; VL07].

GNN models have attracted the interest of the community mainly due to their competitive performance in various tasks, including node, edge, and graph-level predictions [Ham20; Wu+21; Bro+21]. Such models can simultaneously leverage structural and non-structural information (e.g., node features) in the representation learning process. Typically, GNNs learn low-dimensional node representations following a *message passing* mechanism [Gil+17]. Nodes iteratively aggregate neighborhood information, updating their embedding vector and forming the input for the next layer. Let G be a graph with a set of input features $\mathbf{X} \in \mathbb{R}^{N\times F_1} = [\mathbf{x}_1, \dots, \mathbf{x}_N]^\top$. The output of a generic message passing neural network (MPNN) is defined as follows:

$$\mathbf{h}_i^{(l+1)} = \phi\left(\mathbf{h}_i^{(l)}, \bigoplus_{j\in\mathcal{N}_i} \psi\left(\mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)}\right)\right), \tag{4.1}$$

where $\mathbf{H}^{(l)} \in \mathbb{R}^{N \times F_l} = [\mathbf{h}_1^{(l)}, \ldots, \mathbf{h}_N^{(l)}]^\intercal$ is the F_l-dimensional embeddings after l layers such that each $\mathbf{h}_i^{(l)} \in \mathbb{R}^{F_l}$ and $\mathbf{H}^{(1)} = \mathbf{X}$, ψ is a family of learnable message functions, $\bigoplus$ is a permutation-invariant function such as sum, mean, or maximum, and ϕ is an update function. This generic message passing formulation encapsulates several instances of GNN layers. In the case of convolutional networks, a weighted neighborhood aggregation scheme is applied

$$\mathbf{h}_i^{(l+1)} = \phi\left(\mathbf{h}_i^{(l)}, \bigoplus_{j \in \mathcal{N}_i} c_{ij} \psi\left(\mathbf{h}_j^{(l)}\right)\right), \tag{4.2}$$

where the weight c_{ij} is based on the degree of node i and thus can be computed directly from the adjacency matrix $\mathbf{A}$. Prominent models here are the Graph Convolutional Networks (CGN) [KW17] as well as the ChebNet model [DBV16] that follows a spectral formulation. In the case of attention networks proposed in the Graph Attention Network (GAT) model [Vel+18], the importance of each neighborhood node is captured via an attention mechanism [Vas+17a],

$$\mathbf{h}_i^{(l+1)} = \phi\left(\mathbf{h}_i^{(l)}, \bigoplus_{j \in \mathcal{N}_i} \alpha\left(\mathbf{h}_i^{(l-1)}, \mathbf{h}_j^{(l-1)}\right) \psi\left(\mathbf{h}_j^{(l)}\right)\right), \tag{4.3}$$

where α is a learnable self-attention function. Other widely-used GNN instances include GraphSAGE [HYL17a] that comes with a flexible neighborhood aggregation mechanism, and the Graph Isomorphism Network [Xu+19] with a focus on expressiveness.

4.2 TOWARDS DEEP GRAPH NEURAL NETWORKS

> This section is based on material from an article co-authored with Jhony H. Giraldo, Konstantinos Skianis, and Thierry Bouwmans published in the *ACM International Conference on Information and Knowledge Management (CIKM)* [Gir+23].

Although GNNs have demonstrated great success in modeling graph-structured data, they are not immune to the limitations that commonly affect neural networks, such as overfitting and vanishing gradients [Li+19]. In addition, GNNs have two known intrinsic limitations, namely over-smoothing [OS20] and over-squashing [AY21a], that remain poorly understood. These issues arise when stacking multiple message passing layers, leading to the degradation of node representations and the distortion of information aggregated from distant nodes. Over-smoothing results from node features becoming more similar with increasing convolutional layers [OS20], while over-squashing occurs due to severe information compression through bottleneck edges [AY21a]. These issues become particularly relevant when dealing with graphs with large diameters and long-range dependencies between nodes [Dwi+22]. Figure 4.1 shows an example of long-range dependencies between two nodes in a graph. We can quickly realize that the amount of messages we need to send from one node to the other grows exponentially, and this amount of information should be squashed in a few bottleneck edges.

In the related literature, both problems have mainly been addressed separately. For over-smoothing, techniques involving graph topology rewiring, either as a preprocessing step [GWG19] or during training (via dropping edges) [Ron+20], have been proposed. Regarding over-squashing, a fully-adjacent matrix (i.e., each pair of nodes is connected by an edge) is added in the last GNN layer to mitigate the problem [AY21a]. More

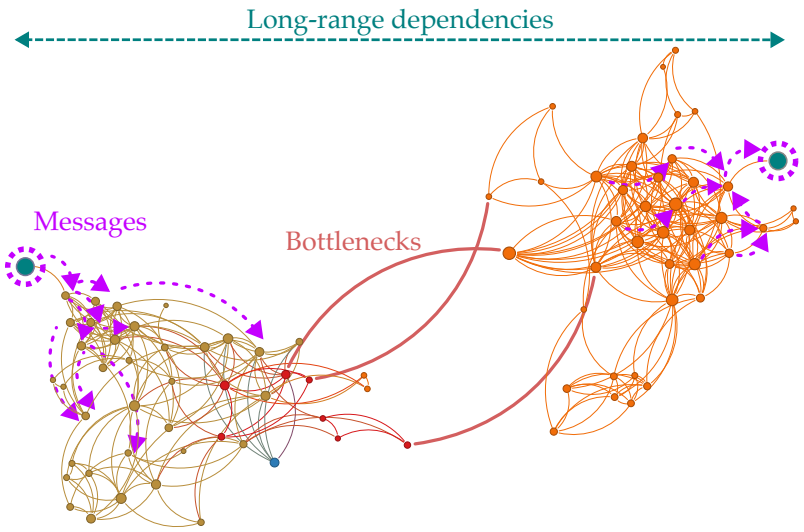


Figure 4.1: Long-range dependencies and bottleneck edges in graph neural networks.

recent approaches consider rewiring methods based on the concept of Ricci curvature in differential geometry [Top+22; Liu+23c]. As we will present shortly, our methodology moves in a similar direction but aims to study both problems simultaneously.

In this section, we establish a fundamental topological relationship between over-smoothing and over-squashing in deep GNNs, leveraging the properties of the random walk matrix and the spectral gap of the Laplacian matrix [Chu97]. Specifically, we employ the Cheeger inequality [Che70] to highlight the inherent trade-off between over-smoothing and over-squashing, emphasizing that improving one aspect invariably deteriorates the other. To navigate this trade-off, we propose an algorithm (called SJLR) that dynamically adds and removes edges during GNN training to mitigate both phenomena.

4.2.1 *The Over-smoothing vs. Over-squashing Trade-off*

OVER-SMOOTHING. Graph convolutions usually use smoothing functions on each layer. Therefore, when we apply several graph convolution layers, the performance can suffer from over-smoothing, where node embeddings from different clusters become mixed up [ZA20; OS20]. Intuitively, we can think of over-smoothing as a random walk transition matrix that is repeatedly applied to a node feature, thus converging to a stationary distribution and washing away all the feature information (this convergence is provided in (4.5)).

OVER-SQUASHING. Over-squashing is a more recent and less understood problem than over-smoothing. As discussed earlier, in graph learning problems that involve long-range dependencies, information from non-adjacent nodes should be propagated through the network without distortion (e.g., Fig. 4.1). Let $\mathcal{B}_r \triangleq \{j \in \mathcal{V} : d_G(i,j) \leq r\}$ be the receptive field of an r-layer GNN, where d_G is the shortest-path distance and $r \in \mathbb{N}$. Let $\partial \mathbf{h}_i^{(r)} / \partial \mathbf{x}_j$ be the *Jacobian* of a node embedding $\mathbf{h}_i^{(r)}$ with respect to some input feature $\mathbf{x}_j$ in node j. Over-squashing can be understood as the inability of $\mathbf{h}_i^{(r)}$ to be affected by $\mathbf{x}_j$ at a distance r. Topping *et al.* [Top+22] showed that $\left|\partial \mathbf{h}_i^{(r+1)} / \partial \mathbf{x}_j\right| \leq (\alpha\beta)^{r+1} \hat{\mathbf{A}}^{r+1}(i,j)$, if $|\nabla \phi_l| \leq \alpha$ and $|\nabla \psi_l| \leq \beta$ for $0 \leq l \leq r$, with ϕ_l, ψ_l differentiable functions. In many graphs, $|\mathcal{B}_r|$ grows

exponentially with r, and then representations of an exponential amount of neighboring nodes should be compressed into fixed-size vectors. For example, if $d_G(i,j) = r+1$ in a binary tree, we have that $\hat{\mathbf{A}}^{r+1}(i,j) = 2^{-1}3^{-r}$, which gives an exponential decay of the node dependence on input features at a distance r [Top+22]. This phenomenon is referred to as over-squashing of information [AY21a; Top+22; DG+23].

OVER-SMOOTHING AND OVER-SQUASHING. Before presenting the trade-off between over-smoothing and over-squashing, let us define two key concepts in the analysis. Let $\mathcal{S} \subset \mathcal{V}$ be a subset of nodes of G. Let $\partial\mathcal{S}$ be the set of edges going from a node in $\mathcal{S}$ to a node in $\mathcal{V} \setminus \mathcal{S}$, i.e., $\partial\mathcal{S} \triangleq \{\{u,v\} \in \mathcal{E} : u \in \mathcal{S}, v \in \mathcal{V} \setminus \mathcal{S}\}$. Therefore, we can define the *Cheeger constant* h_G of G as $h_G \triangleq \min_{\mathcal{S}} h_G(\mathcal{S})$, where $h_G(\mathcal{S}) = |\partial\mathcal{S}| / \min(\mathrm{vol}(\mathcal{S}), \mathrm{vol}(\mathcal{V} \setminus \mathcal{S}))$, and $\mathrm{vol}(\mathcal{S}) = \sum_{i \in \mathcal{S}} d_i$. Intuitively, the Cheeger constant is small when a bottleneck exists in G, i.e., when there are two sets of nodes with few edges between them. Similarly, we know that $h_G > 0$ if G is a connected graph [Chu97]. We can relate the Cheeger constant h_G with the first non-zero eigenvalue of $\mathbf{L}_{\mathrm{sym}}$ through the Cheeger inequality:

$$2h_G \geq \lambda_2 \geq \frac{h_G^2}{2}. \tag{4.4}$$

We notice from (4.4) that to have less bottleneck structure in the graph, we need to promote a large h_G, i.e., having large values of λ_2 will increase h_G since $h_G \geq \lambda_2/2$.

Let us now define $\mathbf{P} = \mathbf{D}^{-1}\mathbf{A}$ to be the random walk transition matrix. For any initial distribution $\mathbf{f} : \mathcal{V} \to \mathbb{R}$ with $\sum_{v \in \mathcal{V}} f(v) = 1$, the distribution after k steps is given by $\mathbf{f}^\top \mathbf{P}^k$, where $\mathbf{f} \in \mathbb{R}^{N \times 1}$ is the vector of initial distributions such that $\mathbf{f}(i)$ is the function evaluated on the ith node. The random walk is *ergodic* when there is a unique stationary distribution $\boldsymbol{\pi}$ satisfying that $\lim_{s \to \infty} \mathbf{f}^\top \mathbf{P}^s = \boldsymbol{\pi}$ [Chu97]. In the case of connected and non-bipartite graphs G, it is known that for an ergodic random walk transition matrix $\mathbf{P}$, the following holds for $s \in \mathbb{N}^+$:

$$\|\mathbf{f}^\top \mathbf{P}^s - \boldsymbol{\pi}\| \leq e^{-s\lambda'} \frac{\max_i \sqrt{d_i}}{\min_j \sqrt{d_j}}, \tag{4.5}$$

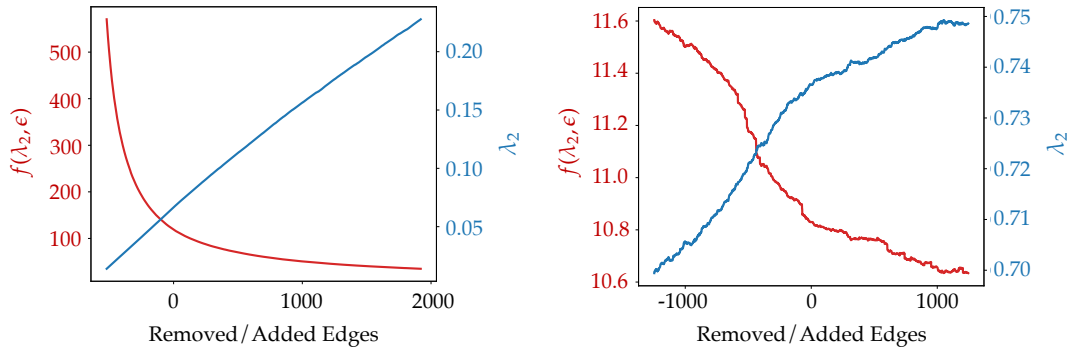
where $\lambda' = \lambda_2$ if $1 - \lambda_2 \geq \lambda_N - 1$, and $2 - \lambda_N$ otherwise [Chu97]. Therefore, we can compute the value of s such that $\|\mathbf{f}^\top \mathbf{P}^s - \boldsymbol{\pi}\| \leq \epsilon$ as follows:

$$s \geq \frac{1}{\lambda' \log\left(\max_i \sqrt{d_i} / \epsilon \min_j \sqrt{d_j}\right)}. \tag{4.6}$$

The key message of (4.5) and (4.6) is a simplified version of the same observations of [Ron+20; OS20], i.e., GNNs converge exponentially to a stationary distribution when stacking several layers. Next, we show that the convergence of this exponential function depends on the spectral gap λ_2. We use this result below to show the underlying relationship between over-smoothing and over-squashing.

Theorem 4.1. *Let h_G be the Cheeger constant of G, and let s be the number of required steps such that the ℓ_2 distance between $\mathbf{f}^\top \mathbf{P}^s$ and $\boldsymbol{\pi}$ is at most ϵ. Therefore, we have that:*

$$2h_G \geq \frac{1}{s} \log\left(\frac{\max_i \sqrt{d_i}}{\epsilon \min_j \sqrt{d_j}}\right). \tag{4.7}$$



(a) Stochastic Block Model with two clusters

(b) Erdős-Rényi graph

Figure 4.2: Mixing steps $f(\lambda_2, \epsilon)$ for $\epsilon = 5 \times 10^{-4}$ vs. number of removed or added edges for a stochastic block model graph with two clusters and an Erdős-Rényi graph.

From Theorem 4.1, we have that if $s \to 0$ then $h_G \to \infty$, i.e., we can reduce the bottlenecks in the graph if we accelerate the convergence to the stationary distribution. Similarly, if $h_G \to 0$, then $s \to \infty$, i.e., we can avoid converging to the stationary distribution if we promote a bottleneck-kind structure in the graph. In other words, we can reduce over-squashing by accelerating the convergence to the stationary distribution that worsens over-smoothing. Correspondingly, we can avoid over-smoothing by promoting having bottlenecks that deteriorate over-squashing.

We can make the connection between over-smoothing and over-squashing more precise using Theorem 4.1, the Simple Graph Convolution (SGC) model[1] [Wu+19], and the developments in [Top+22]. Let $f(\lambda_2, \epsilon) = \frac{1}{\lambda_2} \log\left(\max_i \sqrt{d_i}/\epsilon \min_j \sqrt{d_j}\right)$ be the mixing steps of our graph, i.e., the lower bound in the maximum number of layers of an SGC such that the difference between the initial and stationary distribution is at most ϵ. Figure 4.2 shows, for $\epsilon = 5 \times 10^{-4}$, how $f(\lambda_2, \epsilon)$ and λ_2 change when we add or remove edges in a stochastic block model and an Erdős-Rényi graph. We can increase the mixing steps by removing edges, i.e., we can alleviate over-smoothing by making the graph more "bottleneckness". This partially explains why DropEdge [Ron+20] – a technique that randomly removes edges while training – can alleviate over-smoothing. Conversely, when we increase λ_2 by adding edges as depicted in Fig. 4.2, we encourage greater values of h_G. In other words, we can mitigate over-squashing by reducing the graph's bottleneck effect. This partially explains why rewiring methodologies, such as the one by Topping *et al.* [Top+22], can alleviate over-squashing. However, there is a trade-off between $f(\lambda_2, \epsilon)$ and λ_2 from a topological point of view, i.e., we can increase $f(\lambda_2, \epsilon)$ by removing key edges but λ_2 will decrease, and vice versa. The algorithm to add and remove edges is described next.

4.2.2 *Curvature-based Graph Rewiring Algorithm*

Our rewiring methodology is based on Ollivier's Ricci curvature on graphs [Oll09] and its relationship to the spectral gap. Ricci curvature aims to characterize the average geodesic dispersion at a local neighborhood in the graph. Specifically, curvature captures the property of paths in a given direction to remain "parallel" as in Euclidean space (zero curvature), "converge" as in the surface of a sphere (positive curvature), or "diverge",

1 SGC is a simplified version of the Graph Convolutional Network (GCN) [KW17], where we remove all projection parameters and all non-linear activation functions between layers.

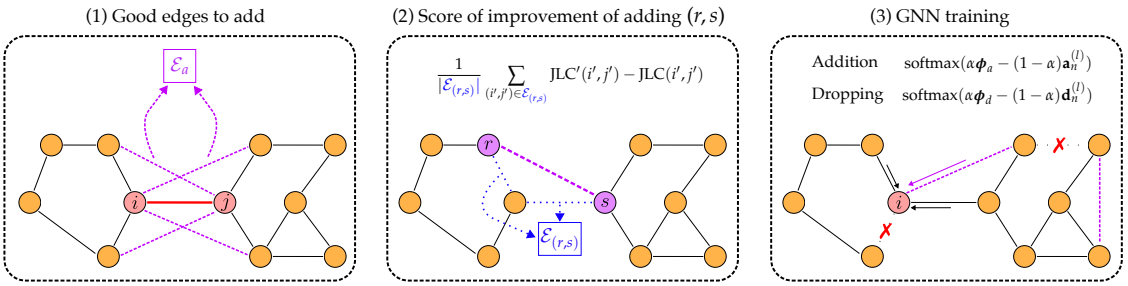


Figure 4.3: The pipeline of the proposed SJLR algorithm. SJLR first saves potential good edges $\mathcal{E}_a$ to add to the graph based on the JLC metric. Secondly, SJLR computes the improvement score of adding these new edges as the average of the JLC improvement that concerns these edges. Thirdly, edges are added and removed according to some probability distribution during the graph neural network training.

which is the case of hyperbolic space (negative curvature) [Top+22; Ngu+23]. Ollivier [Ollo9] considers random walks from nearby nodes $i, j \in \mathcal{V}$ to define the notion of curvature on graphs. In particular, an optimal transport problem is formulated that measures the distance a random walk from node i should travel to meet the random walk from j. Then, curvature $\kappa(i,j)$ is defined as the ratio of the random walk distance and the geodesic (i.e., shortest path) one. According to this definition, if $\kappa(i,j) = 0$, then the random walks from i and j will follow parallel trajectories, maintaining the distance between the two nodes; if $\kappa(i,j) < 0$, the trajectories diverge; and if $\kappa(i,j) > 0$, the trajectories converge. Our analysis uses an easier-to-compute bound of Ollivier's Ricci curvature based on the number of triangles containing two nodes, defined by Jost and Liu [JL14]. For an edge (i,j) in the graph,

$$\mathrm{JLC}(i,j) = -\left(1 - \frac{1}{d_i} - \frac{1}{d_j} - \frac{\#(i,j)}{d_i \wedge d_j}\right)_+ - \left(1 - \frac{1}{d_i} - \frac{1}{d_j} - \frac{\#(i,j)}{d_i \vee d_j}\right)_+ + \frac{\#(i,j)}{d_i \vee d_j}, \tag{4.8}$$

where $\#(i,j)$ is the number of triangles which include (i,j) as nodes, $c_+ \triangleq \max(c,0)$, $c \vee t \triangleq \max(c,t)$, and $c \wedge t \triangleq \min(c,t)$. Besides, we already know that $\kappa(i,j) \geq \mathrm{JLC}(i,j)$, i.e., JLC is a lower bound of Ollivier's Ricci curvature [JL14].

Let us now make the relationship between graph curvature $\kappa(i,j)$ and the analysis presented earlier in Sec. 4.2.1. For a finite graph G, if for any edge (i,j), $\kappa(i,j) \geq \kappa > 0$, then $\lambda_2 \geq \kappa$ [LLY11]. Using the Cheeger inequality in (4.4), we can conclude that if we have positive Ricci curvature everywhere, then $2h_G \geq \kappa$. Therefore, increasing curvature will make the graph to have less bottleneck structure. In other words, having positive JLC ensures that the receptive field of each node in a deep GNN will be polynomial in the hop-distance rather than exponential (see also Corollary 3 in [Top+22]).

Based on this analysis, we introduce the *Stochastic Jost and Liu Curvature Rewiring* (SJLR) algorithm, a GNN agnostic model which uses both curvature information (based on the JLC metric of (4.8)) and node feature information to perform rewiring. The pipeline of the algorithm is shown in Fig. 4.3. In a nutshell, SJLR stochastically adds and removes edges only during training, alleviating over-squashing and over-smoothing without modifying the initial graph, maintaining its original properties. We define a set of hyperparameters for SJLR: (i) let p_A be the percentage of added edges; (ii) let p_D be the percentage of dropped edges; and (iii) let $\alpha \in [0,1]$ be a variable controlling how important is the JLC metric against the embedding information while dropping or adding edges.

SJLR first computes a bank of potential good edges to add, $\mathcal{E}_a$, for improving the JLC curvature (first part in Fig. 4.3), where the JLC metric is calculated (for all $(i,j) \in \mathcal{E}$) and sorted (in ascending order). To this end, we look at every edge (i',j') from the sorted JLC vector, and we compute $\mathcal{A} = \{\{(\mathcal{N}_{i'} \setminus j') \times j'\} \bigcup \{(\mathcal{N}_{j'} \setminus i') \times i'\} : \mathcal{A} \notin \mathcal{E}\}$, which is the set of edges that form triangles with (i',j') and are not in $\mathcal{E}$. We append this set $\mathcal{A}$ to $\mathcal{E}_a$ until we have at least $2p_A|\mathcal{E}|$ edges in $\mathcal{E}_a$. Therefore, we associate a score $\boldsymbol{\sigma}(m)$ to every edge $(r,s) \in \mathcal{E}_a$, which is computed as the average improvement of curvature from adding that edge (r,s) to the graph (second part in Fig. 4.3). Let $\boldsymbol{\sigma} \in \mathbb{R}^{|\mathcal{E}_a|}$ be the vector of JLC improvements, such that:

$$\boldsymbol{\sigma}(m) = \frac{1}{|\mathcal{E}_{(r,s)}|} \sum_{(i',j') \in \mathcal{E}_{(r,s)}} \text{JLC}'(i',j') - \text{JLC}(i',j'), \tag{4.9}$$

where $\text{JLC}'(i',j')$ is the JLC metric of edge (i',j') computed in the augmented graph $G' = (\mathcal{V}, \{\mathcal{E} \cup (r,s)\})$, and $\mathcal{E}_{(r,s)} \subset \mathcal{E}$ is the set of edges that form a triangle with the edge $(r,s) \in \mathcal{E}_a$. Before the GNN training loop, we normalize $\text{JLC}(i,j)$ and $\boldsymbol{\sigma}(m)$ and save them in two vectors ϕ_d and ϕ_a. During the GNN training for each layer l:

(i) We compute the euclidean distances of node features $\mathbf{d}^{(l)}(p) = \|\mathbf{h}_i^{(l)} - \mathbf{h}_j^{(l)}\| \; \forall \; (i,j) \in \mathcal{E}$, $1 \leq p \leq |\mathcal{E}|$ and $\mathbf{a}^{(l)}(q) = \|\mathbf{h}_r^{(l)} - \mathbf{h}_s^{(l)}\| \; \forall \; (r,s) \in \mathcal{E}_a$, $1 \leq q \leq |\mathcal{E}_a|$.

(ii) We normalize $\mathbf{d}^{(l)}$ and $\mathbf{a}^{(l)}$ to be in $[0,1]$ and get $\mathbf{d}_n^{(l)}$ and $\mathbf{a}_n^{(l)}$.

(iii) We drop and add edges according to a probability distribution (third part in Fig. 4.3).

It is worth noting that the addition of potential good edges could be partially parallelized since there is no sequential procedure. This is an important difference regarding previous methods [Top+22], where edges cannot be added in parallel. Let us also mention here that other recent techniques follow a similar approach based on graph curvature optimization to balance over-smoothing and over-squashing [Ngu+23].

4.2.3 *Experimental Evaluation of SJLR*

BASELINE MODELS. We have performed a set of experiments to compare SJLR to eight state-of-the-art methods to alleviate over-smoothing or over-squashing including Residual/Dense Connections (RDC) [Li+19], Graph Diffusion Convolution (GDC) with personalized PageRank kernel [GWG19], DROPEDGE (DE) [Ron+20], PAIRNORM (PN) [ZA20], Differentiable Group Normalization (DGN) [Zho+20], Fully-Adjacent (FA) layers [AY21a], Stochastic Discrete Ricci Flow (SDRF) [Top+22], and First-order Spectral Rewiring (FoSR) [KBM23]. For all experiments, our GNN base models are SGC [Wu+19] and GCN [KW17], tuning the number of layers ($L \in \{2,3,4\}$). Details about the experimental setup can be found in the corresponding article [Gir+23].

DATASETS. We evaluate all methods in nine node classification datasets: *Cornell*, *Texas*, and *Wisconsin* from the WebKB project[2], *Chameleon* [RAS21], *Squirrel* [RAS21], *Actor* [Tan+09], *Cora* [McC+00], *Citeseer* [Sen+08], and *Pubmed* [Nam+12].

NODE CLASSIFICATION RESULTS. Tables 4.1 and 4.2 provide the results for SGC and GCN, respectively. SJLR shows the best overall performance in both cases. We notice

2 http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-11/www/wwkb/

Method	*Cornell*	*Texas*	*Wisconsin*	*Chameleon*	*Squirrel*	*Actor*	*Cora*	*Citeseer*	*Pubmed*
Backbone	53.40$_{\pm 2.11}$	56.69$_{\pm 1.78}$	47.90$_{\pm 1.73}$	38.40$_{\pm 0.69}$	40.52$_{\pm 0.54}$	29.93$_{\pm 0.16}$	76.94$_{\pm 1.31}$	67.45$_{\pm 0.80}$	71.79$_{\pm 2.13}$
GDC	58.65$_{\pm 1.43}$	57.42$_{\pm 0.74}$	45.93$_{\pm 1.05}$	38.13$_{\pm 0.55}$	36.63$_{\pm 0.31}$	**32.25**$_{\pm 0.17}$	76.02$_{\pm 1.70}$	66.22$_{\pm 1.13}$	71.91$_{\pm 2.30}$
DE	61.99$_{\pm 1.04}$	57.88$_{\pm 0.81}$	54.78$_{\pm 0.89}$	40.38$_{\pm 0.47}$	41.28$_{\pm 0.32}$	30.62$_{\pm 0.17}$	80.59$_{\pm 0.80}$	**68.63**$_{\pm 0.51}$	74.47$_{\pm 1.65}$
PN	53.11$_{\pm 1.36}$	50.47$_{\pm 1.04}$	48.72$_{\pm 1.65}$	**41.49**$_{\pm 0.68}$	39.72$_{\pm 0.33}$	22.58$_{\pm 0.29}$	75.55$_{\pm 0.42}$	64.16$_{\pm 0.41}$	73.81$_{\pm 0.52}$
DGN	55.68$_{\pm 1.32}$	57.42$_{\pm 2.59}$	50.67$_{\pm 2.08}$	40.99$_{\pm 0.62}$	41.72$_{\pm 0.29}$	29.53$_{\pm 0.18}$	80.65$_{\pm 0.48}$	67.65$_{\pm 0.59}$	74.95$_{\pm 0.59}$
SDRF	54.68$_{\pm 1.29}$	55.36$_{\pm 1.48}$	47.81$_{\pm 1.51}$	38.07$_{\pm 0.77}$	39.94$_{\pm 0.53}$	30.04$_{\pm 0.17}$	76.04$_{\pm 1.69}$	67.60$_{\pm 0.80}$	69.62$_{\pm 2.35}$
FoSR	53.73$_{\pm 1.75}$	56.33$_{\pm 1.37}$	47.82$_{\pm 2.14}$	38.01$_{\pm 0.73}$	40.68$_{\pm 0.42}$	30.11$_{\pm 0.18}$	78.24$_{\pm 0.98}$	67.04$_{\pm 0.83}$	72.76$_{\pm 2.35}$
SJLR (ours)	**67.37**$_{\pm 1.64}$	**58.40**$_{\pm 1.48}$	**55.42**$_{\pm 0.92}$	40.17$_{\pm 0.49}$	**41.91**$_{\pm 0.34}$	30.81$_{\pm 0.18}$	**81.24**$_{\pm 0.77}$	68.39$_{\pm 0.69}$	**76.28**$_{\pm 0.96}$

Table 4.1: Comparison of SJLR to baseline methods with the SGC model as the backbone.

Method	*Cornell*	*Texas*	*Wisconsin*	*Chameleon*	*Squirrel*	*Actor*	*Cora*	*Citeseer*	*Pubmed*
Backbone	67.34$_{\pm 1.50}$	58.05$_{\pm 0.96}$	52.10$_{\pm 0.95}$	40.35$_{\pm 0.48}$	**42.12**$_{\pm 0.29}$	28.62$_{\pm 0.36}$	81.81$_{\pm 0.26}$	68.35$_{\pm 0.35}$	78.25$_{\pm 0.37}$
RDC	63.78$_{\pm 1.68}$	59.47$_{\pm 1.00}$	50.89$_{\pm 1.00}$	40.33$_{\pm 0.51}$	41.98$_{\pm 0.31}$	28.97$_{\pm 0.33}$	81.54$_{\pm 0.26}$	68.70$_{\pm 0.35}$	78.42$_{\pm 0.39}$
GDC	64.18$_{\pm 1.36}$	56.43$_{\pm 1.15}$	49.61$_{\pm 0.95}$	38.49$_{\pm 0.51}$	33.20$_{\pm 0.29}$	**31.08**$_{\pm 0.27}$	**82.63**$_{\pm 0.23}$	69.15$_{\pm 0.30}$	**79.04**$_{\pm 0.37}$
DE	63.39$_{\pm 1.29}$	57.41$_{\pm 0.93}$	47.84$_{\pm 0.86}$	40.80$_{\pm 0.55}$	41.68$_{\pm 0.39}$	29.99$_{\pm 0.21}$	81.90$_{\pm 0.24}$	68.99$_{\pm 0.36}$	78.53$_{\pm 0.26}$
PN	64.44$_{\pm 1.39}$	**60.93**$_{\pm 1.15}$	51.78$_{\pm 0.95}$	40.37$_{\pm 0.59}$	40.92$_{\pm 0.31}$	28.21$_{\pm 0.21}$	78.89$_{\pm 0.32}$	66.95$_{\pm 0.40}$	76.60$_{\pm 0.41}$
DGN	65.19$_{\pm 1.79}$	58.91$_{\pm 0.93}$	50.76$_{\pm 0.92}$	40.06$_{\pm 0.60}$	41.30$_{\pm 0.32}$	28.32$_{\pm 0.36}$	81.34$_{\pm 0.31}$	69.25$_{\pm 0.35}$	78.06$_{\pm 0.42}$
FA	53.57$_{\pm 0.00}$	59.26$_{\pm 0.00}$	43.02$_{\pm 0.49}$	27.76$_{\pm 0.29}$	31.51$_{\pm 0.00}$	26.69$_{\pm 0.50}$	29.85$_{\pm 0.00}$	23.23$_{\pm 0.00}$	39.24$_{\pm 0.00}$
SDRF	63.88$_{\pm 1.68}$	56.40$_{\pm 0.89}$	40.99$_{\pm 0.62}$	40.74$_{\pm 0.45}$	41.44$_{\pm 0.37}$	28.95$_{\pm 0.33}$	81.42$_{\pm 0.26}$	69.37$_{\pm 0.31}$	77.74$_{\pm 0.42}$
FoSR	56.65$_{\pm 0.93}$	50.01$_{\pm 1.37}$	53.73$_{\pm 1.08}$	40.26$_{\pm 0.50}$	41.83$_{\pm 0.28}$	28.80$_{\pm 0.35}$	81.79$_{\pm 0.26}$	67.99$_{\pm 0.37}$	78.26$_{\pm 0.39}$
SJLR (ours)	**71.75**$_{\pm 1.50}$	60.13$_{\pm 0.89}$	**55.16**$_{\pm 0.95}$	**41.19**$_{\pm 0.46}$	41.86$_{\pm 0.29}$	29.89$_{\pm 0.20}$	81.95$_{\pm 0.25}$	**69.50**$_{\pm 0.33}$	78.60$_{\pm 0.33}$

Table 4.2: Comparison of SJLR to baseline methods with the GCN model as the backbone.

two general trends: (i) rewiring methods such as DE and SJLR dominate in almost all datasets for the experiment with SGC; and (ii) GDC leads in the homophilous datasets *Cora* and *Pubmed* with GCN. Our theoretical results are based on the assumption that there are no non-linear activation functions, so perhaps some nuances are missed for GNNs like GCN. Similarly, we notice that SJLR outperforms SDRF [Top+22] and FoSR [KBM23] in all datasets. For SJLR and SDRF, both methods use the same JLC metric in Tables 4.1 and 4.2, and therefore we are assessing their performance based on how the edges are added or removed. We argue that SJLR is a critical improvement over SDRF regarding the practical adoption of curvature-based methods in GNNs.

ABLATION STUDY. We conduct an ablation study to examine the influence of the addition and removal of edges in SJLR, employing SGC as the backbone model in alignment with our theoretical findings. To explore this, we perform hyperparameter optimization and obtain the results summarized in Table 4.3. The findings suggest that the addition and removal of nodes are complementary and additive in performance. For example, for nearly all datasets, adding and removing edges leads to better performance than applying one strategy alone. This is an important difference regarding recent works [KBM23; Liu+23c] where edges are only added or removed. We theoretically (Theorem 4.1) and empirically (Tables 4.1 and 4.2) show that both, removing and adding edges, are required to find a good compromise in the over-smoothing over-squashing trade-off.

LIMITATIONS. One of the limitations of SJLR is the optimization of hyperparameters. Due to the expanded search space, finding an optimal set of hyperparameters for SJLR through random search becomes more challenging than simpler methods. Another significant limitation is that the current implementation of SJLR relies on a bank $\mathcal{E}_a$ of good edges to add, which is based solely on the triangles of edges with the most negative curvature. As a result, it is not possible to have edges that directly connect long-distance

Dropping	Adding	*Cornell*	*Texas*	*Wisconsin*	*Chameleon*	*Squirrel*	*Actor*	*Cora*	*Citeseer*	*Pubmed*
✗	✗	$53.40_{\pm 2.11}$	$56.69_{\pm 1.78}$	$47.90_{\pm 1.73}$	$38.40_{\pm 0.69}$	$40.52_{\pm 0.54}$	$29.93_{\pm 0.16}$	$76.94_{\pm 1.31}$	$67.45_{\pm 0.80}$	$71.79_{\pm 2.13}$
✓	✗	$59.26_{\pm 1.61}$	$56.16_{\pm 0.85}$	$53.64_{\pm 0.97}$	$39.95_{\pm 0.81}$	$41.27_{\pm 0.39}$	$30.04_{\pm 0.19}$	$80.86_{\pm 0.76}$	$\mathbf{68.48}_{\pm 0.51}$	$75.19_{\pm 1.51}$
✗	✓	$54.07_{\pm 2.36}$	$56.79_{\pm 1.93}$	$45.44_{\pm 1.95}$	$38.20_{\pm 0.58}$	$41.57_{\pm 0.32}$	$29.86_{\pm 0.19}$	$73.90_{\pm 2.01}$	$66.20_{\pm 1.05}$	$69.56_{\pm 2.36}$
✓	✓	$\mathbf{67.37}_{\pm 1.64}$	$\mathbf{58.40}_{\pm 1.48}$	$\mathbf{55.42}_{\pm 0.92}$	$\mathbf{40.17}_{\pm 0.49}$	$\mathbf{41.91}_{\pm 0.34}$	$\mathbf{30.81}_{\pm 0.18}$	$\mathbf{81.24}_{\pm 0.77}$	$68.39_{\pm 0.69}$	$\mathbf{76.28}_{\pm 0.96}$

The best results on each dataset are shown in **bold**.

Table 4.3: Impact of dropping and adding edges in SJLR with the SGC model as the backbone.

nodes. These limitations underscore the need for further research in developing efficient curvature metrics that can effectively account for larger distances in the graph.

4.3 CLUSTERING AND POOLING FOR GRAPH NEURAL NETWORKS

> This section is based on material from an article co-authored with Alexandre Duval published in the *ACM International Conference on Information and Knowledge Management (CIKM)* [DM22].

In this section, we are interested in designing pooling operators used by GNNs to compute graph-level representations, such as in the case of graph classification. Since the goal is to predict the label of the entire graph, standard approaches pool together all nodes' embeddings to create a single graph representation, usually via a simple sum or average operation [Liu+23b]. This *global pooling* discards completely graph structure when computing its final representation, failing to capture the topology of many real-world networks and thus preventing to build effective GNN architectures. More desirable alternatives emerged to solve this limitation. They progressively coarsen the graph between message passing layers, for instance, by regrouping highly connected nodes (i.e., clusters) together into supernodes with adapted adjacency and feature vectors. This allows to better capture the graph's hierarchical structure compared to global pooling without losing relevant information if the coarsening is accurately done. While the first clustering-based pooling algorithms were deterministic [DBV16; Fey+18] – because of their high computational complexity, their transductive nature, and their incapacity to leverage node features – they were replaced by trainable *end-to-end clustering* approaches such as StructPool [YJ20] and DiffPool [Yin+18]. Such methods solve the above limitations, often by learning a cluster assignment matrix along with GNN parameters thanks to a specific loss function, such as a link prediction score.

Despite presenting many advantages, such methods pool nodes together based on simple functions or metrics, often lacking strong supporting theoretical foundations. Besides, they reduce the graph uniquely based on first-order (i.e., edges) information. In many cases, however, graph datasets may not present any clear edge-based connectivity structure, leading to insignificant graph coarsening steps, while nodes might cluster together with respect to more complex (domain-specific) *motifs* [Are+08; Les+09]. Overall, this limits the expressiveness of the hierarchical information captured and, therefore, the classification performance. On top of that, existing pooling operators were surprisingly shown to perform on par with random pooling for many graph classification tasks, raising major concerns [MSK20] and finding limited justifications. This discovery appears rather counter-intuitive as we logically expect the graph coarsening step, that is, the way to pool nodes together, to significantly increase the graph hierarchical information captured in its final representation.

Here, we propose HoscPool, an end-to-end higher-order pooling operator grounded on probabilistic motif spectral clustering to capture a more advanced type of communities

thanks to the incorporation of higher-order connectivity patterns. The latter has shown to be very successful for a wide range of applications [Lee+19] but has not yet been applied to graph classification, while it could greatly benefit from it. Specifically, we hierarchically coarsen the input graph using a cluster assignment matrix learned by defining a well-motivated objective function, which includes continuous relaxations of motif conductance and thus combines various types of connectivity patterns for greater expressiveness. Since the process is fully differentiable, we can stack several such pooling layers, intertwined by message passing layers, to capture graph hierarchical information.

4.3.1 *Graph Cut and Motif Conductance*

NORMALIZED CUT. Clustering involves partitioning the vertices of a graph into K disjoint subsets with more intra-cluster than inter-cluster edges [VL07; For10; MV13a]. One of the most common and effective ways to formulate this task is by solving the Normalized Cut problem [SM00]:

$$\min_{\mathcal{S}_1,\dots,\mathcal{S}_K} \sum_{k=1}^{K} \frac{\text{cut}(\mathcal{S}_k, \bar{\mathcal{S}}_k)}{\text{vol}(\mathcal{S}_k)}, \tag{4.10}$$

where $\bar{\mathcal{S}}_k = \mathcal{V} \setminus \mathcal{S}_k$, $\text{cut}(\mathcal{S}_k, \bar{\mathcal{S}}_k) = \sum_{i \in \mathcal{S}_K, j \in \bar{\mathcal{S}}_K} \mathbf{A}_{ij}$, and $\text{vol}(\mathcal{S}_k) = \sum_{i \in \mathcal{S}_k, j \in \mathcal{V}} \mathbf{A}_{ij}$ (for simplicity in the notation, we use $\mathbf{A}_{ij}$ instead of $\mathbf{A}(i,j)$). Unlike the simple min-cut objective, (4.10) scales each term by the cluster volume, thus enforcing clusters to be "reasonably large" and avoiding degenerate solutions where most nodes are assigned to a single cluster. Although minimizing (4.10) is NP-hard, there are approximation algorithms with theoretical guarantees for finding clusters with small conductance, such as spectral clustering, which proposes clusters determined based on the eigenvalue decomposition of the Laplacian matrix [VL07].

MOTIF CONDUCTANCE. Higher-order connectivity patterns, also known as motifs, i.e., small network subgraphs like triangles $\blacktriangledown$, are known to be the fundamental building blocks of complex networks [Mil+02; Bat+20]. While the Normalized Cut builds on first-order connectivity patterns (i.e., edges), various alternative formulations have been proposed to cluster a network based on specific higher-order substructures [BGL16; TPM17]. Formally, for graph G, motif M made of $|M|$ nodes, and $\mathcal{M} = \{\mathbf{v} \in \mathcal{V}^{|M|} | \mathbf{v} = M\}$ the set of all instances of M in G, they propose to search for the partition $\mathcal{S}_1, \dots, \mathcal{S}_K$ minimizing motif conductance:

$$\min_{\mathcal{S}_1,\dots,\mathcal{S}_K} \sum_{k=1}^{K} \frac{\text{cut}_M^{(G)}(\mathcal{S}_k, \bar{\mathcal{S}}_k)}{\text{vol}_M^{(G)}(\mathcal{S}_k)}, \tag{4.11}$$

where $\text{cut}_M^{(G)}(\mathcal{S}_k, \bar{\mathcal{S}}_k) = \sum_{\mathbf{v} \in \mathcal{M}} \mathbf{1}(\exists i, j \in \mathbf{v} | i \in \mathcal{S}_k, j \in \bar{\mathcal{S}}_k)$, i.e., the number of instances $\mathbf{v}$ of M with at least one node in $\mathcal{S}_k$ and at least one node in $\bar{\mathcal{S}}_k$; and $\text{vol}_M^{(G)}(\mathcal{S}_k) = \sum_{\mathbf{v} \in \mathcal{M}} \sum_{i \in \mathbf{v}} \mathbf{1}(i \in \mathcal{S}_k)$, i.e., the number of motif instance endpoints in $\mathcal{S}_k$. Let us note that besides motif conductance, the modularity criterion for community detection has also been extended based on motifs [Are+08].

We now introduce the motif adjacency matrix $\mathbf{A}_M$, where each entry $(\mathbf{A}_M)_{ij}$ represents the number of motifs in which both node i and node j participate. Its diagonal has zero values. Formally, $(\mathbf{A}_M)_{ij} = \sum_{\mathbf{v} \in \mathcal{M}} \mathbf{1}(i, j \in \mathbf{v}, i \neq j)$. G_M is the graph induced by $\mathbf{A}_M$.

$(\mathbf{D}_M)_{ii} = \sum_{j=1}^{N}(\mathbf{A}_M)_{ij}$ and $\mathbf{L}_M$ are the motif degree and motif Laplacian matrices. Now, we define a discrete cluster assignment matrix $\mathbf{S} \in \{0,1\}^{N\times K}$ where $\mathbf{S}_{ij} = 1$ if $v_i \in \mathcal{S}_j$ and 0 otherwise. We denote by $\mathbf{S}_j = [S_{1j}, \ldots, S_{Nj}]^\top$ the j^{th} column of $\mathbf{S}$, which indicates the nodes belonging to cluster $\mathcal{S}_j$. Based on that, we can express (4.11) as

$$\begin{aligned} &\max_{\mathbf{S}\in\{0,1\}^{N\times K}} \sum_{k=1}^{K} \frac{\sum_{i,j\in\mathcal{V}}(\mathbf{A}_M)_{ij}\mathbf{S}_{ik}\mathbf{S}_{jk}}{\sum_{i,j\in\mathcal{V}}\mathbf{S}_{ik}(\mathbf{A}_M)_{ij}} \\ \equiv &\max_{\mathbf{S}\in\{0,1\}^{N\times K}} \sum_{k=1}^{K} \frac{\mathbf{S}_k^\top \mathbf{A}_M \mathbf{S}_k}{\mathbf{S}_k^\top \mathbf{D}_M \mathbf{S}_k} \equiv \min_{\mathbf{S}\in\{0,1\}^{N\times K}} -\mathrm{Tr}\left(\frac{\mathbf{S}^\top \mathbf{A}_M \mathbf{S}}{\mathbf{S}^\top \mathbf{D}_M \mathbf{S}}\right), \end{aligned} \tag{4.12}$$

where the division sign in the last line is an element-wise division on the diagonal of both matrices. By definition, $\mathbf{S}$ is subject to the constraint $\mathbf{S}\mathbf{1}_K = \mathbf{1}_N$, i.e., each node belongs exactly to one cluster. To make the optimization problem feasible, we relax it to a probabilistic framework, where $\mathbf{S}$ take continuous values in the range $[0,1]$, representing cluster membership probabilities, i.e., each entry $\mathbf{S}_{ik}$ denotes the probability that node i belongs to cluster k. Next, we will demonstrate how it can be optimized within any GNN model.

4.3.2 *Higher-order Hierarchical Clustering and Pooling*

HIGHER-ORDER CLUSTERING WITH GNNS. We aim to design a trainable cluster assignment matrix $\mathbf{S}$ that learns to find relevant clusters based on higher-order connectivity patterns in an end-to-end manner within any GNN architecture. Such an approach can address the limitations of (motif) spectral clustering: we cluster nodes based both on graph topology and node features; leverage higher-order connectivity patterns; avoid the expensive eigenvalue decomposition of the motif Laplacian matrix; and allow to cluster out-of-sample graphs. We compute the soft cluster assignment matrix $\mathbf{S}$ using one (or more) fully connected (FC) layer(s), mapping each node's representation $\mathbf{X}_{i*}$ to its probabilistic cluster assignment vector $\mathbf{S}_{i*}$. We apply a softmax activation function to enforce the constraint inherited from (4.12): $\mathbf{S}_{ij} \in [0,1]$ and $\mathbf{S}\mathbf{1}_K = \mathbf{1}_N$:

$$\mathbf{S} = \mathrm{FC}(\mathbf{X}; \boldsymbol{\Theta}). \tag{4.13}$$

$\boldsymbol{\Theta}$ are trainable parameters, optimized by minimizing the unsupervised loss function $\mathcal{L}_{mc}$, which approximates the relaxed formulation of the motif conductance problem (4.12):

$$\mathcal{L}_{mc} = -\frac{1}{K} \cdot \mathrm{Tr}\left(\frac{\mathbf{S}^\top \mathbf{A}_M \mathbf{S}}{\mathbf{S}^\top \mathbf{D}_M \mathbf{S}}\right). \tag{4.14}$$

Referring to the spectral clustering formulation, $\mathcal{L}_{mc} \in [-1, 0]$. It reaches -1 when $\mathcal{G}_M$ has $\geq K$ connected components (no motif endpoints are separated by clustering), and zero when for each pair of nodes participating in the same motif (i.e., $(\mathbf{A}_M)_{ij} > 0$), the cluster assignments are orthogonal: $\langle \mathbf{S}_{i*}, \mathbf{S}_{j*} \rangle = 0$.

In fact, we allow the combination of several motifs inside our objective function (4.14) via $\mathcal{L}_{mc} = \sum_j \alpha_j \mathcal{L}^j_{mc}$ where $\mathcal{L}^j_{mc}$ denotes the objective function with respect to a particular motif type j (e.g., edge , triangle  , 4-nodes cycle ) and α_j is an importance factor. This also increases the power of our method, allowing us to find communities of nodes w.r.t. a hierarchy of higher-order substructures. As a result, the graph coarsening step will pool together more relevant groups of nodes, potentially capturing informative patterns

in subsequent layers, ultimately producing richer graph representation. We implement it for edge and triangle motifs:

$$\mathcal{L}_{mc} = -\frac{\alpha_1}{K} \cdot \mathrm{Tr}\left(\frac{\mathbf{S}^\top \mathbf{A}\mathbf{S}}{\mathbf{S}^\top \mathbf{D}\mathbf{S}}\right) - \frac{\alpha_2}{K} \cdot \mathrm{Tr}\left(\frac{\mathbf{S}^\top \mathbf{A}_M\mathbf{S}}{\mathbf{S}^\top \mathbf{D}_M\mathbf{S}}\right). \tag{4.15}$$

We let α_1 and α_2 to be dynamic functions of the epoch, subject to $\alpha_1 + \alpha_2 = 1$, allowing to first optimize higher-order motifs before moving on to smaller ones. It helps refine the level of granularity progressively and was found desirable empirically. This is the higher-order clustering formulation that we consider in the paper. In practice, we enforce more rigorously the hard cluster assignment through an auxiliary orthogonality loss function $\mathcal{L}_o$, which encourages more balanced and discrete clusters. More details about that can be found in the paper [DM22].

Similarly to other cluster-based pooling operators, our method relies on two assumptions. Firstly, nodes are identifiable via their features. Secondly, node features represent a good initialization for computing cluster assignments. The latter is realistic due to the homophily property of many real-world networks as well as the smoothing effect of message passing layers discussed in Sec. 4.2, which render connected nodes more similar.

HIGHER-ORDER GRAPH COARSENING WITH HOSCPOOL. The methodology detailed in the previous sections is a general clustering technique that can be used for any clustering task on any graph dataset. In this part, we utilize it to form a pooling operator called HOSCPOOL, which exploits the cluster assignment matrix $\mathbf{S}$ to generate a coarsened version of the graph (with fewer nodes and edges) that preserves critical information and embeds higher-order connectivity patterns. More precisely, it coarsens the existing graph by creating super-nodes from the derived clusters, with a new edge set and feature vector, depending on previous nodes belonging to this cluster. More formally,

$$\text{HOSCPOOL} : G = (\mathbf{X}, \mathbf{A}) \to G^{pool} = (\mathbf{X}^{pool}, \mathbf{A}^{pool})$$
$$\mathbf{A}^{pool} = \mathbf{S}^\top \mathbf{A}\mathbf{S} \quad \text{and} \quad \mathbf{X}^{pool} = \mathbf{S}^\top \mathbf{X}.$$

Each entry $\mathbf{X}^{pool}_{i,j}$ denotes feature j's value for cluster i, calculated as a sum of feature j's value for the nodes belonging to cluster i, weighted by the corresponding cluster assignment scores. $\mathbf{A}^{pool} \in \mathbb{R}^{K \times K}$ is a symmetric matrix where $\mathbf{A}^{pool}_{i,j}$ can be viewed as the connection strength between cluster i and cluster j. Given our optimization function, it will be a diagonal-dominant matrix, which will hamper the propagation across adjacent nodes. For this reason, we remove self-loops. We also symmetrically normalize the new adjacency matrix. Lastly, note that we use the original $\mathbf{A}$ and $\mathbf{X}$ for this graph coarsening step; their motif counterparts $\mathbf{A}_M$ and $\mathbf{X}_M$ are simply leveraged to compute the loss function. Our work thus differs clearly from diffusion methods and traditional GNNs leveraging higher-order.

Because our GNN-based implementation of motif spectral clustering is fully differentiable, we can stack several HOSCPOOL layers, intertwined with message passing layers, to hierarchically coarsen the graph representation. Ultimately, a global pooling and some dense layers produce a graph prediction. The parameters of each HOSCPOOL layer can be learned end-to-end by jointly optimizing:

$$\mathcal{L} = \mathcal{L}_{mc} + \mu \mathcal{L}_o + \mathcal{L}_s, \tag{4.16}$$

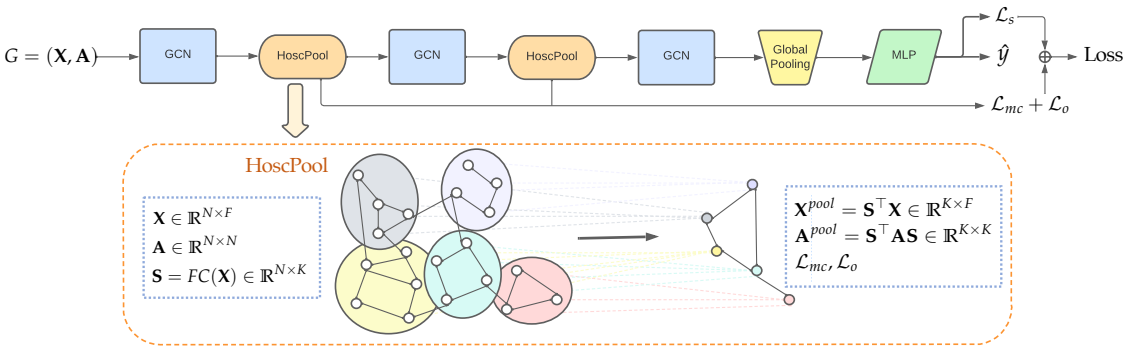


Figure 4.4: A graph classification pipeline with HoscPool hierarchical pooling.

where $\mathcal{L}_s$ denotes any supervised loss for a particular downstream task (here, the cross entropy loss). This way, we should be able to hierarchically capture relevant graph higher-order structure while learning GNN parameters to ultimately better classify the graphs within our dataset. A schematic representation of a graph classification pipeline with HoscPool is given in Fig. 4.4.

4.3.3 *Experimental Evaluation of HoscPool*

In this section, we provide an overview of the experimental evaluation of HoscPool, focusing on its performance as (i) a differentiable higher-order clustering algorithm and (ii) a hierarchical pooling operator for graph classification.

GRAPH CLUSTERING. For this experiment, we first run a message passing (MP) layer; in this case a GCN model with skip connection for initial features: $\bar{\mathbf{X}} = \text{ReLU}(\mathbf{A}\mathbf{X}\boldsymbol{\Theta}_1 + \mathbf{X}\boldsymbol{\Theta}_2)$, where $\boldsymbol{\Theta}_1$ and $\boldsymbol{\Theta}_2$ are trainable weight matrices. We then run a Multi-Layer Perceptron (MLP) to produce the cluster assignment matrix of dimension $N \times K$, trained end-to-end by optimizing the unsupervised loss function $\mathcal{L}_{mc} + \mu\mathcal{L}_o$.

In terms of benchmark graphs, we use a collection of node classification datasets with ground truth community labels: citation networks *Cora, PubMed*; collaboration networks *DBLP, Coauthor CS*; co-purchase networks *Amazon Photo, Amazon PC*; communication networks *Polblogs* and *Eu-email*[3]. We have also constructed three synthetic datasets: *Syn1*, *Syn2*, *Syn3* (based on several random graphs) where node labels are determined based on higher-order community structure and node features are simple graph statistics. They are designed to show the effectiveness of HoscPool when datasets have clear higher-order structure, which is not always the case for the standard baseline datasets chosen.

We have compared HoscPool with the original spectral clustering (SC) and with motif spectral clustering (MSC) based on motif conductance instead of edge conductance, i.e., SC applied on $\mathbf{A}_M$ [BGL16]. We also consider pooling baselines DiffPool [Yin+18] and MinCutPool [BGA20]. We refer to all methods by their pooling name for simplicity, although this experiment focuses on the clustering part and does not involve the coarsening step. For ablation study, let HoscPool-1 and HoscPool-2 denote HoscPool where $\mathcal{L}_{mc}$ in (4.15) has $\alpha_2 = 0$ (first-order connectivity only) and $\alpha_1 = 0$ (higher-order only), respectively.

3 All datasets are available in Pytorch Geometric.

Dataset	SC	MSC	DiffPool	MinCutPool	HoscPool-1	HoscPool-2	HoscPool
Cora	$0.150_{\pm 0.002}$	$0.056_{\pm 0.014}$	$0.308_{\pm 0.023}$	$0.391_{\pm 0.028}$	$0.435_{\pm 0.032}$	$\underline{0.464}_{\pm 0.036}$	$\mathbf{0.502}_{\pm 0.029}$
PubMed	$0.183_{\pm 0.002}$	$0.002_{\pm 0.000}$	$0.098_{\pm 0.006}$	$0.214_{\pm 0.066}$	$\underline{0.230}_{\pm 0.071}$	$0.215_{\pm 0.073}$	$\mathbf{0.260}_{\pm 0.054}$
Photo	$\underline{0.592}_{\pm 0.008}$	$0.451_{\pm 0.011}$	$0.171_{\pm 0.004}$	$0.086_{\pm 0.014}$	$0.495_{\pm 0.068}$	$0.513_{\pm 0.083}$	$\mathbf{0.598}_{\pm 0.101}$
PC	$0.464_{\pm 0.002}$	$0.166_{\pm 0.009}$	$0.043_{\pm 0.008}$	$0.026_{\pm 0.006}$	$0.497_{\pm 0.040}$	$\underline{0.499}_{\pm 0.036}$	$\mathbf{0.528}_{\pm 0.041}$
CS	$0.273_{\pm 0.006}$	$0.011_{\pm 0.009}$	$0.383_{\pm 0.048}$	$0.431_{\pm 0.060}$	$0.479_{\pm 0.022}$	$\underline{0.701}_{\pm 0.029}$	$\mathbf{0.731}_{\pm 0.018}$
DBLP	$0.027_{\pm 0.003}$	$0.005_{\pm 0.006}$	$0.186_{\pm 0.014}$	$\mathbf{0.334}_{\pm 0.026}$	$\underline{0.326}_{\pm 0.027}$	$0.284_{\pm 0.026}$	$0.312_{\pm 0.027}$
Polblogs	$0.017_{\pm 0.000}$	$0.014_{\pm 0.001}$	$0.317_{\pm 0.010}$	$0.440_{\pm 0.390}$	$\underline{0.992}_{\pm 0.003}$	$\mathbf{0.994}_{\pm 0.001}$	$\mathbf{0.994}_{\pm 0.005}$
Email-eu	$0.485_{\pm 0.030}$	$0.382_{\pm 0.019}$	$0.096_{\pm 0.034}$	$0.253_{\pm 0.028}$	$0.317_{\pm 0.026}$	$\mathbf{0.488}_{\pm 0.025}$	$\underline{0.476}_{\pm 0.021}$
Syn1	$0.000_{\pm 0.000}$	$\mathbf{1.000}_{\pm 0.000}$	$0.035_{\pm 0.000}$	$0.043_{\pm 0.008}$	$0.041_{\pm 0.006}$	$\mathbf{1.000}_{\pm 0.000}$	$\mathbf{1.000}_{\pm 0.000}$
Syn2	$0.003_{\pm 0.000}$	$0.050_{\pm 0.003}$	$0.081_{\pm 0.008}$	$0.902_{\pm 0.028}$	$0.942_{\pm 0.028}$	$\mathbf{1.000}_{\pm 0.000}$	$\mathbf{1.000}_{\pm 0.000}$
Syn3	$\mathbf{1.000}_{\pm 0.000}$	$\mathbf{1.000}_{\pm 0.000}$	$0.067_{\pm 0.001}$	$0.052_{\pm 0.002}$	$0.115_{\pm 0.006}$	$\underline{0.826}_{\pm 0.005}$	$\mathbf{1.000}_{\pm 0.000}$

Table 4.4: NMI obtained by clustering the nodes of various networks over ten different runs.

Table 4.4 shows the experimental results using the Normalized Mutual Information (NMI) clustering metric [For10; MV13a]. HoscPool has a competitive performance compared to the baseline models across most datasets. This trend is emphasized in synthetic datasets, where we know that higher-order structure is critical, proving the benefits of our clustering method. We have observed that DiffPool often fails to converge to a good solution. Besides, MinCutPool, as also discussed in [Tsi+23], sometimes get stuck in degenerate solutions (e.g., *Amazon PC* and *Photo* – all nodes are assigned to less than 10% of clusters), failing to converge even when tuning model architecture and hyper-parameters. HoscPool-1 shows superior performance and alleviates this issue, meaning that it can be considered as an improved version of MinCutPool. MSC often performs badly, revealing its excessive dependence on the presence of motifs. On the contrary, our results highlight the robustness of HoscPool to the limited presence of motifs due to its consideration for node features. Besides, HoscPool's attention to finer granularity levels allows to group nodes primarily based on motifs while still considering edges when necessary, which may be the reason for the performance improvement with respect to HoscPool-2. This ablation study proves the relevance of our underlying claims: incorporating higher-order information leads to better communities, and combining several motifs further helps. In terms of efficiency, the main complexity of HoscPool lies in the derivation of $\mathbf{A}_M$, which remains relatively fast for triangle motifs: $\mathbf{A}_M = \mathbf{A}^2 \odot \mathbf{A}$. Despite being slower to compute compared to other coarsening graph techniques such as MinCutPool, it is still affordable even for the larger graphs considered here.

GRAPH CLASSIFICATION. For this task, we consider a fixed network architecture composed of: GNN – Pooling – GNN – Pooling – GNN – Global Pooling – Dense ($\times 2$). We sometimes add skip connections and global pooling to the output of the first and second GNN; and concatenate the resulting vector to the third GNN's output. A pooling block produces a cluster assignment matrix of dimension $num\_nodes \times \text{int}(num\_nodes \times 0.25)$.

We have used several common benchmark datasets for graph classification, taken from TUDataset [Mor+20], including three bioinformatics protein datasets *Proteins*, *Enzymes*, and *D&D*; one mutagen *Mutagenicity*; one anticancer activity dataset *NCI1*; two chemical compound datasets *Cox-2-MD*, *ER-MD*; one social network *Reddit-Binary*. *Bench-hard* is taken from this source[4] where $\mathbf{X}$ and $\mathbf{A}$ are completely uninformative if considered alone. We split them into a training set (80%), validation set (10%), and test set (10%). For featureless graphs, we use constant features.

4 https://github.com/FilippoMB/Benchmark_dataset_for_graph_classification

Method	*Proteins*	*NCI1*	*Mutagen.*	*DD*	*Reddit-B*	*Cox2-MD*	*ER-MD*	*b-hard*
NoPool	$71.6_{\pm 4.1}$	$77.1_{\pm 1.9}$	$78.1_{\pm 1.3}$	$71.2_{\pm 2.2}$	$80.1_{\pm 2.6}$	$58.7_{\pm 3.2}$	$72.2_{\pm 2.9}$	$66.5_{\pm 0.5}$
Random	$75.7_{\pm 3.2}$	$77.0_{\pm 1.7}$	$79.2_{\pm 1.3}$	$77.1_{\pm 1.5}$	$89.3_{\pm 2.6}$	$62.9_{\pm 3.6}$	$73.0_{\pm 4.5}$	$69.1_{\pm 2.1}$
GMT	$75.0_{\pm 4.2}$	$74.9_{\pm 4.3}$	$79.4_{\pm 2.2}$	$78.1_{\pm 3.2}$	$86.7_{\pm 2.6}$	$58.9_{\pm 3.6}$	$74.3_{\pm 4.5}$	$70.1_{\pm 3.4}$
MinCutPool	$75.9_{\pm 2.4}$	$76.8_{\pm 1.6}$	$78.6_{\pm 1.8}$	$78.4_{\pm 2.8}$	$89.0_{\pm 1.4}$	$58.9_{\pm 5.1}$	$75.5_{\pm 4.0}$	$72.6_{\pm 1.5}$
DiffPool	$73.8_{\pm 3.7}$	$76.7_{\pm 2.1}$	$77.9_{\pm 2.3}$	$76.3_{\pm 2.1}$	$87.3_{\pm 2.4}$	$57.1_{\pm 4.8}$	$76.8_{\pm 4.8}$	$70.7_{\pm 2.0}$
EigPool	$74.2_{\pm 3.1}$	$75.0_{\pm 2.2}$	$75.2_{\pm 2.7}$	$75.1_{\pm 1.8}$	$82.8_{\pm 2.1}$	$59.8_{\pm 3.4}$	$73.1_{\pm 3.8}$	$69.1_{\pm 3.1}$
SAGPool	$70.6_{\pm 3.5}$	$74.1_{\pm 3.9}$	$74.4_{\pm 2.7}$	$71.5_{\pm 4.1}$	$74.7_{\pm 4.5}$	$56.9_{\pm 9.7}$	$71.7_{\pm 8.2}$	$39.6_{\pm 9.6}$
ASAP	$74.4_{\pm 2.6}$	$74.3_{\pm 1.6}$	$76.8_{\pm 2.4}$	$73.2_{\pm 2.5}$	$84.1_{\pm 1.1}$	$60.5_{\pm 5.5}$	$74.5_{\pm 5.9}$	$70.5_{\pm 1.7}$
HoscPool-1	$76.7_{\pm 2.5}$	$77.3_{\pm 1.6}$	$79.8_{\pm 1.6}$	$\underline{78.8}_{\pm 2.0}$	$91.2_{\pm 1.0}$	$61.6_{\pm 3.5}$	$76.2_{\pm 4.2}$	$72.4_{\pm 0.8}$
HoscPool-2	$\underline{77.0}_{\pm 3.1}$	$\mathbf{80.3}_{\pm 2.0}$	$\underline{92.8}_{\pm 1.5}$	$\mathbf{66.4}_{\pm 4.6}$	$\mathbf{92.8}_{\pm 1.5}$	$\mathbf{66.4}_{\pm 4.6}$	$\underline{77.9}_{\pm 4.3}$	$\underline{73.5}_{\pm 0.8}$
HoscPool	$\mathbf{77.5}_{\pm 2.3}$	$\underline{79.9}_{\pm 1.7}$	$\mathbf{82.3}_{\pm 1.3}$	$\mathbf{79.4}_{\pm 1.8}$	$\mathbf{93.6}_{\pm 0.9}$	$\underline{64.6}_{\pm 3.9}$	$\mathbf{78.2}_{\pm 3.8}$	$\mathbf{74.0}_{\pm 0.4}$

Table 4.5: Graph classification accuracy of various pooling operators.

We have compared HoscPool to representative graph classification baseline models, involving pooling operators DiffPool [Yin+18], MinCutPool [BGA20], EigPool [Ma+19], SAGPool [LLK19], ASAP [RST20], and GMT [BKH21]. We implement a random pooling operator (Random) to assess the benefits of pooling similar nodes together and a model with a single global pooling operator (NoPool) to assess how useful leveraging hierarchical information is.

The graph classification results are reported in Table 4.5, from which we draw the following conclusions. First of all, we observe that performing pooling proves useful, contrary to NoPool, in most cases. HoscPool compares favorably on all datasets w.r.t. other pooling baselines. Higher-order connectivity patterns are more desirable than first-order ones, and combining both is even better. This observation is aligned with the findings of the previous paragraph and shows that better clustering (i.e., graph coarsening) is correlated with better classification performance. However, while the clustering performance of HoscPool is significantly better than baselines, the performance gap has slightly closed down on this task. Even more surprising, the benefits of existing advanced node-grouping or node-dropping methods are not considerable with respect to the Random pooling baseline.

DISCUSSION. From the experiments conducted here, we have noticed that despite effectively learning a cluster assignment matrix – that assigns nodes to more clusters and better balances the number of nodes per cluster – the performance gain w.r.t. the Random baseline model is often not significant. To explain this behavior, we have examined the properties of the graph datasets used. The experiments are detailed in our article [DM22]. In a nutshell, the benchmark graphs are relatively small, with few node types co-existing in the same graph, weak homophily, and a relatively poor community structure, which clustering algorithms aim to exploit. Besides, because most datasets do not have dense node features (only labels), the node identifiability assumption is shaken and does not enable our MLP of (4.13) to fully distinguish between same-label-nodes, thus making it impossible to place them in distinct clusters. On top of that, we now need to learn a clustering pattern that extends to all graphs, which is a much more complex task (compared to a single graph in the clustering task). All these points raise questions for future work regarding the functioning of hierarchical pooling operators for graph-level prediction tasks.

4.4 EXPLAINABILITY IN GRAPH NEURAL NETWORKS

This section is based on material from an article co-authored with Alexandre Duval published in the *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)* [DM21].

In this section, we investigate the aspect of explainability in Graph Neural Networks. Similarly to traditional deep learning frameworks, GNNs showcase a complex functioning that is rather opaque to humans. As the field grows, understanding them becomes essential, especially for safety-critical applications [Yua+23]. While there exist a variety of explanation methods [SGK17; Sel+17; Gui+18], they are not well suited for geometric data as they fall short in their ability to incorporate graph topology information. Preliminary approaches have proposed extensions to GNNs, including decomposition methods that distribute the prediction score among input features using the weights of the network architecture through backpropagation. However, in addition to limited performance, they require the model's internal knowledge and show gradient saturation issues due to the discrete nature of the adjacency matrix [Pop+19]. Another strategy concerns the design of perturbation methods, including GNNExplainer [Yin+19], PGExplainer [Luo+20], and GraphMask [SCT21] which monitor variations in model prediction concerning different input perturbations. Regarding other approaches, surrogate models such as GraphLIME [Hua+23] and PGM-Explainer [VT20] build on the LIME framework [RSG16]. They approximate the black box GNN model locally by learning an interpretable model on a dataset built around the instance of interest. Rule-based methods such as Discern [VF+22] extract activation rules that capture co-activated neurons in a prediction task. Those rules are further interpreted through representative graphs embedded in the subspace defined by the rule. Finally, XGNN [Yua+20] produces model-level insights via graph generation trained using reinforcement learning

In this section, we first introduce a unified explanation framework encapsulating recently proposed explainers for GNNs. It not only serves as a connecting force between them but also provides a different and common view of their functioning, which should inspire future work. Then, we propose GraphSVX, an explanation model that carefully constructs and combines the key components of the unified pipeline to jointly capture the average marginal contribution of node features and graph nodes towards the explained prediction. We show that GraphSVX ultimately computes, via an efficient algorithm, the game theoretic aspect of Shapley values [Sha53; LL17].

4.4.1 *A Unified Framework for GNN Explainers*

We first present a unified framework to regroup several GNN explanation methods, as well as to motivate our proposed methodology, which will be introduced in the next section. The key differences across most explanation models lie in the definition and optimization of the three main blocks of the pipeline, as shown in Fig. 4.5:

- Mask generates discrete or continuous masks over features $\mathbf{M}_F \in \mathbb{R}^F$, nodes $\mathbf{M}_N \in \mathbb{R}^N$, and edges $\mathbf{M}_E \in \mathbb{R}^{N \times N}$, according to a specific strategy.
- Gen outputs a new graph $G' = (\mathbf{X}', \mathbf{A}')$ from the masks $(\mathbf{M}_E, \mathbf{M}_N, \mathbf{M}_F)$ and the original graph $G = (\mathbf{X}, \mathbf{A})$.
- Expl generates explanations, often offered as a vector or a graph, using a function g whose definition vary across baselines.

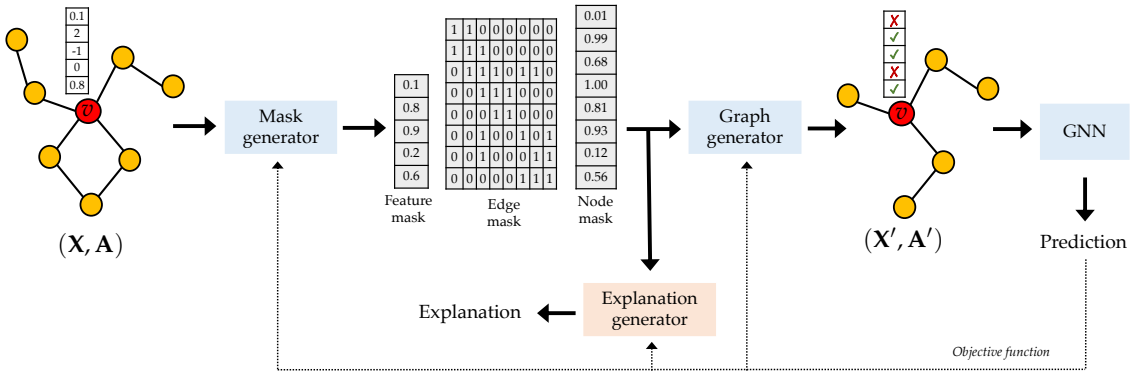


Figure 4.5: Overview of unified framework. All methods take as input a given graph $G = (\mathbf{X}, \mathbf{A})$, feed it to a mask generator (MASK) to create three masks over nodes, edges, and features. These masks are then passed to a graph generator (GEN) that converts them to the original input space $(\mathbf{X}', \mathbf{A}')$ before feeding them to the original GNN model f. The resulting prediction $f(\mathbf{X}', \mathbf{A}')$ is used to improve the mask generator, the graph generator or the downstream explanation generator (EXPL), which ultimately provides the desired explanation—using masks and $f(\mathbf{X}', \mathbf{A}')$. This passage through the framework is repeated many times to create a proper dataset $\mathcal{D}$ from which each generator block learns. Usually, only one is optimized with a carefully defined process involving the new and original GNN predictions.

In the following, we show how two baseline models, namely GNNEXPLAINER and PGEXPLAINER, fit the pipeline. The formulation has been applied to several other models discussed in our article [DM21]. $\odot$ stands for the element-wise multiplication operation, σ is the softmax function, $||$ the concatenation operation, and $\mathbf{M}^{\text{ext}}$ describes the extended vector $\mathbf{M}$ with repeated entries, whose size makes the operation feasible. All three masks are not considered for a single method; some are ignored as they have no effect on final explanations. Indeed, one often studies node feature $\mathbf{M}_F$ or graph structure (via $\mathbf{M}_E$ or $\mathbf{M}_N$).

GNNEXPLAINER's key component is the mask generator. It generates both $\mathbf{M}_F$ and $\mathbf{M}_E$, where $\mathbf{M}_E$ has discrete values and $\mathbf{M}_F$ continuous ones. They are both randomly initialized and jointly optimized via a mutual information loss function $MI(Y, (\mathbf{M}_E, \mathbf{M}_F)) = H(Y) - H(Y|\mathbf{A}', \mathbf{X}')$, where GEN gives $\mathbf{A}' = \mathbf{A} \odot \sigma(\mathbf{M}_E)$ and $\mathbf{X}' = \mathbf{X} \odot \mathbf{M}_F^{\text{ext}}$. Y represents the class label and $H(\cdot)$ the entropy term. EXPL simply returns the learned masks as explanations via the identity function $g(\mathbf{M}_E, \mathbf{M}_F) = (\mathbf{M}_E, \mathbf{M}_F)$.

PGEXPLAINER is very similar to GNNEXPLAINER. MASK generates only an edge mask $\mathbf{M}_E$ using a multi-layer neural network MLP_ψ and the learned matrix $\mathbf{Z}$ of node representations: $\mathbf{M}_E = \text{MLP}_\psi(G, \mathbf{Z})$. The new graph is constructed with $\text{GEN}(\mathbf{X}, \mathbf{A}, \mathbf{M}_E) = (\mathbf{X}, \mathbf{A} \odot \rho(\mathbf{M}_E))$, where ρ denotes a reparametrization trick. The obtained prediction $f_v(\mathbf{X}, \mathbf{A}')$ is also used to maximize mutual information with $f_v(\mathbf{X}, \mathbf{A})$ and backpropagates the result to optimize MASK. As for GNNEXPLAINER, EXPL provides $\mathbf{M}_E$ as explanations.

As we will see in the next section, the proposed GRAPHSVX model carefully exploits the potential of this framework through a better design and combination of complex mask, graph, and explanation generators—in the perspective of improving performance and embedding desirable properties in explanations.

4.4.2 *Shapley Value Explanations for Graph Neural Networks*

GraphSVX is a post hoc model-agnostic explanation method specifically designed for GNNs that jointly computes graph structure and node feature explanations for a single instance. More precisely, GraphSVX constructs a perturbed dataset made of binary masks for nodes and features $(\mathbf{M}_N, \mathbf{M}_F)$, and computes their marginal contribution $f(\mathbf{X}', \mathbf{A}')$ towards the prediction using a graph generator $\textsc{Gen}(\mathbf{X}, \mathbf{A}, \mathbf{M}_F, \mathbf{M}_N) = (\mathbf{X}', \mathbf{A}')$. It then learns a carefully defined explanation model on the dataset $(\mathbf{M}_N||\mathbf{M}_F, f(\mathbf{X}', \mathbf{A}'))$ and provides it as an explanation. Ultimately, it produces a unique deterministic explanation that decomposes the original prediction and has a real signification (Shapley values) and other desirable properties. Without loss of generality, we consider a node classification task to present the method.

THE SHAPLEY VALUE. Before we present the proposed methodology, let us briefly introduce the concept of the Shapley value. The Shapley value is a method from Game Theory that describes how to fairly distribute the total gains of a game to the players depending on their respective contributions, assuming they all collaborate. It is obtained by computing the average marginal contribution of each player when added to any possible coalition of players [Sha53]. This method has been extended to explain machine learning model predictions on tabular data [LC01; SK10], assuming that each feature of the explained instance ($\mathbf{x}$) is a player in a game where the prediction is the payout.

The characteristic function $val : S \to \mathbb{R}$ captures the marginal contribution of the coalition $S \subseteq \{1, \dots, F\}$ of features towards the prediction $f(\mathbf{x})$ with respect to the average prediction: $val(S) = \mathbb{E}[f(\mathbf{X})|\mathbf{X}_S = \mathbf{x}_s] - \mathbb{E}[f(\mathbf{X})]$. We isolate the effect of a feature j via $val(S \cup \{j\}) - val(S)$ and average it over all possible ordered coalitions S to obtain its Shapley value as:

$$\phi_j(val) = \sum_{S \subseteq \{1,\dots,F\} \setminus \{j\}} \frac{|S|!\ (F - |S| - 1)!}{F!} \big(val(S \cup \{j\}) - val(S)\big). \tag{4.17}$$

The notion of fairness is defined by four axioms (*efficiency*, *dummy*, *symmetry*, *additivity*), and the Shapley value is the unique solution satisfying them. In practice, the sum becomes impossible to compute because the number of possible coalitions (2^{F-1}) increases exponentially by adding more features. We thus approximate Shapley values using sampling [LL17].

MASK AND GRAPH GENERATORS. Let us now continue with the presentation of the proposed GraphSVX model. First, we create an efficient mask generator algorithm that constructs discrete feature and node masks, respectively denoted by $\mathbf{M}_F \in \{0,1\}^F$ and $\mathbf{M}_N \in \{0,1\}^N$. Intuitively, for the explained instance v, we aim at studying the joint influence of a subset of features and neighbors of v towards the associated prediction $f_v(\mathbf{X}, \mathbf{A})$. The mask generator helps us determine the subset being studied. Associating one with a variable (node or feature) means it is considered, while zero means it is discarded. For now, we let Mask randomly sample from all possible (2^{F+N-1}) pairs of masks $\mathbf{M}_F$ and $\mathbf{M}_N$, meaning all possible coalitions S of features and nodes (v is not considered in explanations). Let $\mathbf{z}$ be the random variable accounting for selected variables, $\mathbf{z} = (\mathbf{M}_F \| \mathbf{M}_N)$. This can be considered a simplified version of the true mask generator.

We now would like to estimate the joint effect of this group of variables towards the original prediction. We thus isolate the effect of selected variables marginalized over excluded ones and observe the change in prediction. We define GEN : $(\mathbf{X}, \mathbf{A}, \mathbf{M}_F, \mathbf{M}_N) \to (\mathbf{X}', \mathbf{A}')$, which converts the obtained masks to the original input space, in this perspective. Due to the message passing scheme of GNNs, studying the influence of nodes and features jointly is tricky. Unlike GNNEXPLAINER, we avoid overlapping effects by considering feature values of v (instead of the whole subgraph around v) and all nodes except v. Several options are possible to cancel out a node's influence on the prediction, such as replacing its feature vector with random or expected values. Here, we decide to isolate the node in the graph, which totally removes its effect on the prediction. Similarly, to neutralize the effect of a feature, as GNNs do not handle missing values, we set it to the dataset's expected value. Formally, it translates into:

$$\mathbf{X}' = \mathbf{X} \;\text{ with }\; \mathbf{X}'_v = \mathbf{M}_F \odot \mathbf{X}_v + (\mathbf{1} - \mathbf{M}_F) \odot \boldsymbol{\mu} \tag{4.18}$$

$$\mathbf{A}' = (\mathbf{M}_N^{\text{ext}\top} \cdot \mathbf{A} \cdot \mathbf{M}_N^{\text{ext}}) \odot I(\mathbf{A}), \tag{4.19}$$

where $\boldsymbol{\mu} = (\mathbb{E}[\mathbf{X}(:,1)], \ldots, \mathbb{E}[\mathbf{X}(:,F)])^\top$ and $I(\cdot)$ captures the indirect effect of k-hop neighbours of v ($k > 1$), which is often underestimated. Indeed, if a 3-hop neighbor w is considered alone in a coalition, it becomes disconnected from v in the new graph G'. This prevents us from capturing its indirect impact on the prediction since it does not pass information to v. To remedy this problem, we select one shortest path $\mathcal{P}$ connecting w to v and include $\mathcal{P}$ back in the new graph. To keep the influence of the new nodes (in $\mathcal{P} \setminus \{w, v\}$) switched off, we set their feature vector to mean values. To finalize the perturbation dataset, we pass $\mathbf{z}' = (\mathbf{X}', \mathbf{A}')$ to the GNN model f and store each sample $(\mathbf{z}, f(\mathbf{z}'))$ in a dataset $\mathcal{D}$. $\mathcal{D}$ associates with a subset of nodes and features of v their estimated influence on the original prediction.

EXPLANATION GENERATOR. Here, we build a surrogate model g on the dataset $\mathcal{D} = \{(\mathbf{z}, f(\mathbf{z}'))\}$ and provide it as explanation. More rigorously, an explanation ϕ of f is normally drawn from a set of possible explanations, called interpretable domain Ω. It is the solution of the following optimization process: $\phi = \arg\min_{g \in \Omega} \mathcal{L}_f(g)$, where the loss function attributes a score to each explanation. The choice of Ω has a large impact on the type and quality of the obtained explanation. We choose broadly Ω as the set of interpretable models and, more precisely, the set of Weighted Linear Regression (WLR). In short, we intend our model to learn to calculate the individual effect of each variable towards the original prediction from the joint effect $f(\mathbf{z}')$, using many different coalitions S of nodes and features. This is made possible by the definition of the input dataset $\mathcal{D}$ and is enforced by a cross-entropy loss function:

$$\begin{aligned} \mathcal{L}_{f,\boldsymbol{\pi}}(g) &= \sum_{\mathbf{z}} \big(g(\mathbf{z}) - f(\mathbf{z}')\big)^2 \boldsymbol{\pi}_{\mathbf{z}}, \\ \text{where} \qquad \boldsymbol{\pi}_{\mathbf{z}} &= \frac{F+N-1}{(F+N)\cdot|\mathbf{z}|} \cdot \binom{F+N-1}{|\mathbf{z}|}^{-1}. \end{aligned} \tag{4.20}$$

$\boldsymbol{\pi}$ is a kernel weight that attributes a high weight to samples $\mathbf{z}$ with small or large dimensions, or in different terms, groups of features and nodes with few or many elements—since it is easier to capture individual effects from the combined effect in these cases. In the end, we provide the learned parameters of g as an explanation. Each coefficient corresponds to a node of the graph or a feature of v and represents its

estimated influence on the prediction $f_v(\mathbf{X}, \mathbf{A})$. In fact, it approximates the extension of the Shapley value to graphs, as shown in the next paragraph.

DECOMPOSITION MODEL. We first justify why extending the Shapley value to graphs is relevant. Looking back at the original theory, each player contributing to the total gain is allocated a proportion of that gain depending on its fair contribution. Since a GNN model prediction is fully determined by node feature information ($\mathbf{X}$) and graph structural information ($\mathbf{A}$), both edges/nodes and node features are players that should be considered in explanations. In practice, we have redefined how to capture the influence of players (features and nodes) towards the prediction as $val(S) = \mathbb{E}_{\mathbf{X}_v}[f_v(\mathbf{X}, \mathbf{A}_S)|\mathbf{X}_{vS} = \mathbf{x}_{vS}] - \mathbb{E}[f_v(\mathbf{X}, \mathbf{A})]$. $\mathbf{A}_S$ is the adjacency matrix where all nodes in $\overline{S}$ (not in S) have been isolated.

Assuming model linearity and feature independence, we show that GRAPHSVX, in fact, captures via $f(\mathbf{z}')$ the marginal contribution of each coalition S towards the prediction:

$$\begin{aligned}
\mathbb{E}_{\mathbf{X}_v}[f_v(\mathbf{X}, \mathbf{A}_S)|\mathbf{X}_{vS}] &= \mathbb{E}_{\mathbf{X}_{v\overline{S}}|\mathbf{X}_{vS}}[f_v(\mathbf{X}, \mathbf{A}_S)] \\
&\approx \mathbb{E}_{\mathbf{X}_{v\overline{S}}}[f_v(\mathbf{X}, \mathbf{A}_S)] && \text{by independence} \\
&\approx f_v(\mathbb{E}_{\mathbf{X}_{v\overline{S}}}[\mathbf{X}], \mathbf{A}_S) && \text{by linearity} \\
&= f_v(\mathbf{X}', \mathbf{A}'),
\end{aligned}$$

where $\mathbf{A}' = \mathbf{A}_S$ and $\mathbf{X}'(i,j) = \begin{cases} \mathbb{E}[\mathbf{X}(:,j)] \text{ if } i = v \text{ and } j \in \overline{S} \\ \mathbf{X}(i,j) \text{ otherwise.} \end{cases}$

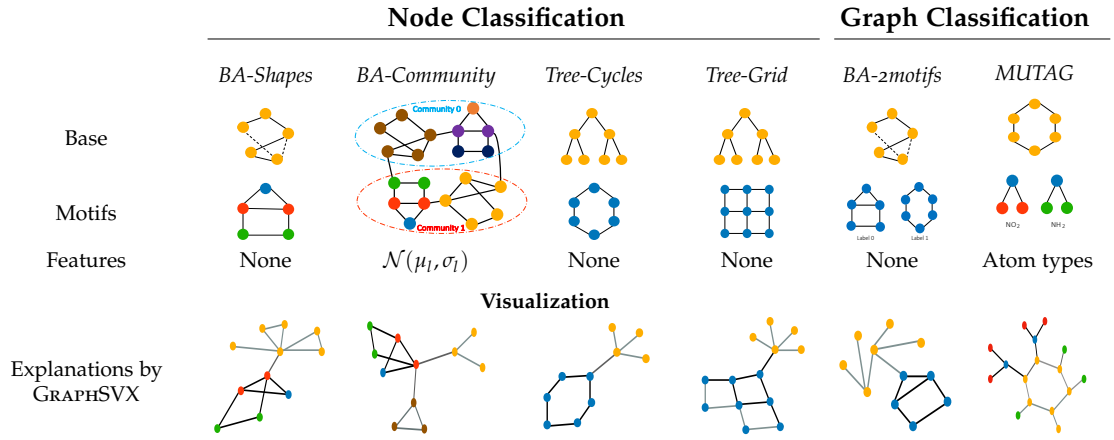
Using the above, we prove that GRAPHSVX calculates the Shapley values on graph data. This builds on the fact that Shapley values can be expressed as an additive feature attribution model, as shown by [LL17] in the case of tabular data. In this perspective, we set π_v such that $\pi_v(\mathbf{z}) \to \infty$ when $|\mathbf{z}| \in \{0, F+N\}$ to enforce the *efficiency* axiom: $g(\mathbf{1}) = f_v(\mathbf{X}, \mathbf{A}) = \mathbb{E}[f_v(\mathbf{X}, \mathbf{A})] + \sum_{i=1}^{F+N} \phi_i$. This holds due to the specific definition of GEN and g (i.e., EXPL), where $g(\mathbf{1}) = f_v(\mathbf{X}, \mathbf{A})$ and the constant ϕ_0, also called base value, equals $\mathbb{E}_{\mathbf{X}_v}[f_v(\mathbf{X}, \mathbf{A}_v)] \approx \mathbb{E}[f_v(\mathbf{X}, \mathbf{A})]$, so the mean model prediction. $\mathbf{A}_v$ refers to $\mathbf{A}_\emptyset$, where v is isolated.

4.4.3 *Experimental Evaluation of GraphSVX*

We assess the effectiveness of GRAPHSVX on node and graph classification tasks, both in the presence of ground truth explanations as well as on complex real-world datasets without ground truth, by testing the model's ability to filter noisy features and noisy nodes from explanations.

BASELINE MODELS. We have compared the performance of GRAPHSVX to baseline models that incorporate graph structure in explanations, namely GNNEXPLAINER [Yin+19], PGEXPLAINER [Luo+20], PGM-EXPLAINER [VT20], GRAPHLIME [Hua+23], and XGNN [Yua+20].

DATASETS WITH GROUND TRUTH. For node classification, we consider synthetic graphs, where each input graph is a combination of a base graph together with a set of motifs, which differ across datasets [Luo+20; Yin+19]. The label of each node is determined based on its belonging and role in the motif. Consequently, the explanation



Explanation Accuracy						
GNNExplainer	0.83	0.75	0.86	0.84	0.68	0.65
PGM-Explainer	0.96	0.92	0.95	0.87	0.91	0.72
PGExplainer	0.92	0.81	0.96	0.88	0.85	**0.79**
GraphSVX	**0.99**	**0.93**	**0.97**	**0.93**	**0.99**	0.77

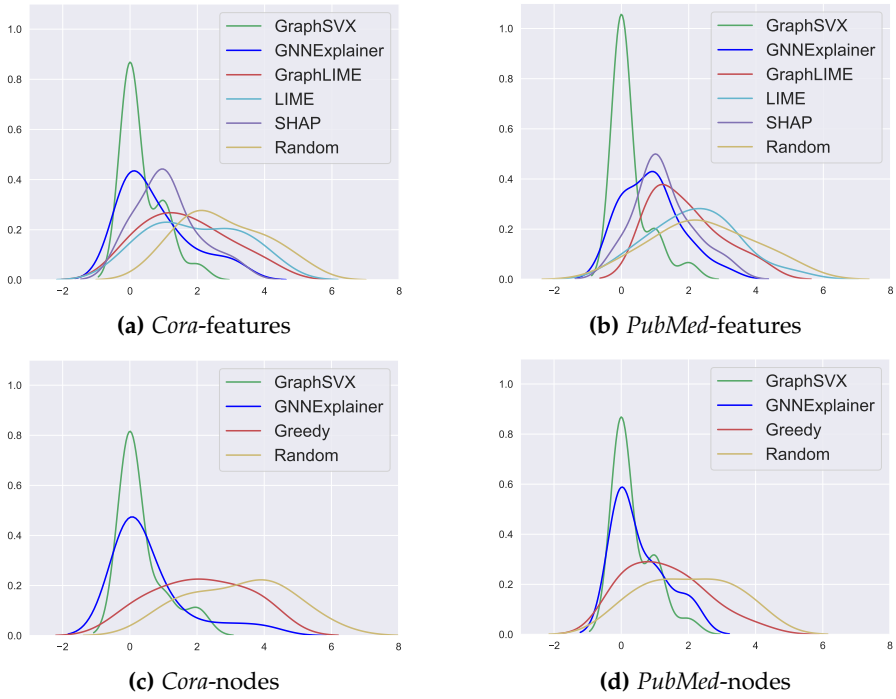
Table 4.6: Evaluation of GraphSVX and baseline GNN explainers on various datasets. The top part describes the construction of each dataset, with its base graph, the motif added, and the node features generated. Node labels are represented by colors. Then, we provide a visualization of GraphSVX's explanations, where an important substructure is drawn in bold, as well as a quantitative evaluation based on the accuracy metric.

for a node in a motif should be the nodes in the same motif, which creates a ground truth explanation. This ground truth can be used to measure the performance of an explainer via an accuracy metric. In the case of graph classification, similar to above, we use a synthetic dataset called *BA-2motifs*. We also consider *MUTAG* consisting of molecular graphs, each assigned to one of two classes based on its mutagenic effect.

We train the same GNN model in all cases [KW17]. The performance is measured with an accuracy metric (node or edge accuracy depending on the nature of explanations) on top-k explanations, where k equals the ground truth dimension. We formalize the evaluation as a binary classification of nodes (or edges) where nodes (or edges) inside motifs are positive and the rest ones negative.

The results on both synthetic and real-life datasets are summarized in Table 4.6. As shown visually and quantitatively, GraphSVX correctly identifies essential graph structure, outperforming the leading baselines on all but one task and offering higher theoretical guarantees and human-friendly explanations. On *MUTAG*, the special nature of the dataset and ground truth favors edge explanation methods, which capture slightly more information than node explainers. In terms of efficiency, our explainer is slower than the scalable PGExplainer despite our efficient approximation but is often comparable to GNNExplainer (running time experiments are provided in the corresponding article [DM21]).

REAL-WORLD DATASETS WITHOUT GROUND TRUTH. We evaluate GraphSVX on two real-world datasets (citation networks) without ground truth explanations: *Cora* and *PubMed*. Instead of looking if the explainer provides the correct explanation, we check that it does not provide a bad one. In particular, we introduce noisy features and nodes to the dataset, train a new GNN on the latter (which we verify does not leverage these



(a) *Cora*-features **(b)** *PubMed*-features

(c) *Cora*-nodes **(d)** *PubMed*-nodes

Figure 4.6: Frequency distributions of noisy features (a), (b) and nodes (c), (d) using a GAT model on *Cora* and *PubMed*.

noisy variables), and observe if our explainer includes them in explanations. Our aim is to investigate whether GraphSVX is capable of filtering out irrelevant features and nodes from complex datasets and selecting only the relevant information for its explanations.

We artificially add 20% new "noisy" features to the dataset, defining them using the existing features' distribution. We re-train a 2-layer GAT model [Vel+18] on this noisy data, whose test accuracy is above 75%. We then produce explanations for 50 test samples using different explainer baseline models on *Cora* and *PubMed*, and we compare their performance by assessing how many noisy features are included in explanations among top-k features. Ultimately, we compare the resulting frequency distributions using a kernel density estimator (KDE). Intuitively, since features are noisy, they are not used by the GNN model and thus are unimportant. Therefore, the less noisy features are included in the explanation, the better the explainer.

In the evaluation, we also include the well-known SHAP [LL17] and LIME [RSG16] models. We also consider a Greedy procedure, which greedily removes the most contributory features/nodes of the prediction until the prediction changes, and a Random one, which randomly selects k features/nodes as the explanations for the prediction being explained. The results are depicted in Fig. 4.6 (a)-(b). For all GNNs and on all datasets, the number of noisy features selected by GraphSVX is close to zero and, in general, lower than existing baselines—demonstrating its robustness to noise.

We follow a similar idea for noisy neighbors instead of noisy features, where the dataset's distribution determines each new node's connectivity and feature vector. Only a few baselines (GNNExplainer, Greedy, Random) among the ones selected previously can be included for this task since GraphLIME, SHAP, and LIME do not provide explanations for nodes. As before, this evaluation builds on the assumption that a well-performing model will not consider as essential these noisy variables. We check the validity of this assumption for the GAT model by looking at its attention weights.

We retrieve the average attention weight of each node across the different GAT layers and compare the one attributed to noisy nodes versus normal nodes. We expect it to be lower for noisy nodes, which proves to be true: 0.11 vs. 0.15. As shown in Fig. 4.6 (c)-(d), GRAPHSVX also outperforms all baselines, showing nearly no noisy nodes in explanations. GNNEXPLAINER achieves almost as good performance on both datasets (and in several evaluation settings).

4.5 DISCUSSION

In this chapter, we investigated aspects related to the design and functioning of graph neural network models. We first studied the trade-off between over-smoothing and over-squashing in the design of deep GNNs, proposing SJLR to mitigate both phenomena. Nevertheless, our curvature-based metric relies only on local topological information. Besides, the algorithm indirectly considers node features (contrary to directly injecting them into the curvature metric). We then introduced HOSCPOOL, a hierarchical graph coarsening model for clustering and pooling in GNNs. Despite its good performance, our experiments indicated that further investigation is needed regarding the design of pooling operators that will significantly improve performance on graph classification tasks. This point is closely related to the underlying topological properties of graph classification datasets and their impact on the performance of hierarchical graph coarsening algorithms. Finally, we introduced GRAPHSVX, a model to explain the predictions of a GNN. An interesting direction here would be to consider how such models can be extended in the case of multilayer graphs studied in Chapter 3.

5

GRAPH REPRESENTATION LEARNING FOR INFLUENCE MAXIMIZATION

The goal of this chapter is to examine applications of graph representation learning models in the tasks of influence learning and maximization. The chapter begins with an overview of the influence maximization problem, emphasizing how machine learning techniques could be leveraged (Sec. 5.1). Then, we introduce two methodologies aiming to address different challenges. First, we discuss IMINFECTOR, an algorithm that uses embeddings learned from diffusion cascades to perform model-independent influence maximization (Sec. 5.2). Then, we present GLIE, a graph neural network that learns to estimate the influence spread and further utilize it for influence maximization (Sec. 5.3). The material of this chapter is based on the following publications:

- George Panagopoulos, Fragkiskos D. Malliaros, and Michalis Vazirgiannis. Influence Maximization Using Influence and Susceptibility Embeddings. In *ICWSM*,. 2020, pp. 511–521.
- George Panagopoulos, Fragkiskos D. Malliaros, and Michalis Vazirgiannis. Multi-Task Learning for Influence Estimation and Maximization. *IEEE Trans. Knowl. Data Eng.* 34:9 (2022), pp. 4398–4409.
- George Panagopoulos, Nikolaos Tziortziotis, Michalis Vazirgiannis, and Fragkiskos D. Malliaros. Maximizing Influence with Graph Neural Networks. In *ASONAM*,. 2023.

5.1 BACKGROUND

Social influence governs multiple aspects of our lives. From choosing the product you buy and the restaurant you visit to adopting political ideas, the strength of interactions with others and the way information propagates can be decisive factors in a person's life. In the real world, it can be used for epidemic containment [DOT14], echo chamber detection [Min+22], and misinformation mitigation in social networks [BAEA11; Far+17]. Formally, social influence is defined as a directed measure between two users and represents how possible it is for the target user to adapt the behavior or copy the action of the source user. In viral marketing, influence is used to simulate how information flows through the network toward finding the optimal set of nodes to start a campaign from, the well-known *influence maximization* problem [KKT03].

In influence maximization, the users are represented as nodes in the graph while the edges depict a relationship, such as who-follows-whom or friendship, and are associated with an influence probability p. A stochastic diffusion model, such as the *independent cascade* model [KKT03], simulates how influence spreads over the graph. It is used to compute the number of users activated during a diffusion simulation, called the *influence spread*. The aim is to find the optimal set of k users to maximize the influence spread of a diffusion cascade starting from them [Li+18; KKT03]. More formally, according to the independent cascade model, the influence spread starts with a set of initially activated

nodes, often referred to as the seed set $\mathcal{S}$. In each discrete time step, an active node has a chance to activate its inactive neighbors based on the associated influence probabilities. This process continues iteratively until no more nodes can be influenced. The final set of activated nodes constitutes the influence spread $\sigma(\mathcal{S})$, i.e., the average number of nodes reached by the seed set. Notice, however, that due to the stochastic nature of the model, Monte Carlo sampling is used to estimate the influence spread of a particular seed set, which directly impacts the time complexity.

Given a graph $G = (\mathcal{V}, \mathcal{E})$, a diffusion model, and a positive integer k, the influence maximization problem aims to select a subset $\mathcal{S}^* \subseteq \mathcal{V}$ composed of k seed nodes that maximize the influence spread, i.e., $\sigma(\mathcal{S}^*) = \text{argmax}_{\mathcal{S}\subseteq\mathcal{V}}\, \sigma(\mathcal{S})$. Influence maximization is an NP-Hard problem under the independent cascade and other related models. Although the optimal solution cannot be computed in polynomial time, approximate solutions can be used due to two key properties satisfied by $\sigma(\cdot)$. Specifically, let us consider an influence function $\sigma(\cdot)$ and subsets $\mathcal{S}$ and $\mathcal{T}$ such that $\mathcal{S} \subseteq \mathcal{T} \subseteq \mathcal{V}$. Then, $\sigma(\cdot)$ is monotone if $\sigma(\mathcal{S}) \leq \sigma(\mathcal{T})$ and submodular if $\sigma(S \cup \{v\}) - \sigma(S) \geq \sigma(T \cup \{v\}) - \sigma(T)$. Intuitively, monotonicity implies that adding more nodes to a seed set $\mathcal{S}$ does not harm its influence spread, while the submodularity can be understood as diminishing marginal gains of the influence spread [KKT03; Li+18].

Due to the monotonicity and submodularity properties of the influence function $\sigma(\cdot)$, the optimal solution can be approximated through a greedy hill-climbing algorithm. The Greedy algorithm starts with an empty seed set $\mathcal{S}$, in which k nodes are iteratively added, maximizing the marginal gain, i.e., $\sigma(v|\mathcal{S}) = \sigma(\mathcal{S} \cup \{v\}) - \sigma(\mathcal{S})$. In terms of theoretical guarantees about the quality of the solution, the Greedy algorithm provides a lower bound on the influence spread of the selected seed set $\mathcal{S}$ which is $\sigma(\mathcal{S}) \geq (1 - 1/e)\sigma(\mathcal{S}^*)$, where $\mathcal{S}^*$ corresponds to the optimal solution [KKT03]. In the related literature, several approaches have been introduced to improve the running time of the Greedy algorithms, retaining or not the theoretical guarantees. Examples include the Celf [Les+07] and Celf++ [GLL11a] algorithms that exploit the submodularity property of influence spread to reduce the number of Monte Carlo simulations. Moreover, let us mention that different instances of the problem have been studied from diverse communities [MM15; PRF23; Kit+10; Che+12]. We refer the reader to relevant articles for a more detailed presentation [Li+18; Li+23].

There have also been attempts to address influence maximization with learned influence parameters from real past cascades [GBL11; GBL10]. These models, however, suffer from overfitting due to the number of parameters, which is proportional to the number of edges. To reduce the number of parameters, a more practical approach is to express the influence probabilities as a combination of the nodes' influence and susceptibility embeddings learned from the cascades [BLG16]. More recently, influence learning methods devoid of diffusion models have been proposed. Such models learn embeddings based on co-occurrence in cascades [Fen+18], similar to node embedding techniques studied in Chapter 2. Nevertheless, most of these influence learning techniques have not been used directly in the influence maximization problem. Although more accurate in diffusion prediction, the input of such models consists of node-context pairs derived from the propagation network of the cascade, a realization of the underlying network (e.g., follow edges) based on time-precedence in the observed cascade (e.g., retweets). However, constructing this propagation network is a time-consuming process. The models presented in this chapter aim to address some of these challenges. A detailed exploration of machine learning techniques for influence maximization tasks can be found in related survey articles and tutorials [Li+23; PM21].

5.2 LEARNING EMBEDDINGS FOR INFLUENCE ESTIMATION AND MAXIMIZATION

This section is based on material from two articles co-authored with George Panagopoulos and Michalis Vazirgiannis published in the *International AAAI Conference on Web and Social Media (ICWSM)* [PMV20] and *IEEE Trans. Knowl. Data Eng. (TKDE)* [PMV22].

One of the main problems in the influence maximization literature is that the diffusion models utilize random or uniform influence parameters. Experiments have shown that such an approach produces a less realistic influence spread than models with empirical parameters [AD18]. Even when the parameters are learned empirically from historical logs of diffusion cascades, the independent cascade (IC) model is utilized [Du+13; BLG16], which is problematic for two reasons. Apart from severe overfitting due to the massive number of parameters, this approach assumes influence independence throughout related nodes or edges of the same node. This assumption overlooks the network's assortativity, meaning that an influential node is more prone to affect a susceptible node than a less influential node, even when the edge of the latter to the susceptible is stronger than the edge of the former. In addition, diffusion models themselves suffer from oversimplifying assumptions overlooking several characteristics of real cascades [PMM18], which can provide inaccurate estimations compared to actual cascades. At the same time, they are highly sensitive to their parameters [Du+14].

To tackle these issues, this section presents IMINFECTOR (*Influence Maximization with Influencer Vectors*), a unified approach that uses representations acquired from diffusion cascades, enabling model-independent influence maximization that scales in real-world datasets. The first part of the methodology is composed of a multi-task neural network that learns embeddings for nodes that initiate cascades (influencer vectors) and for those that participate in them (susceptible vectors). Our focus on the initiators of cascades is justified by empirical evidence on the predominantly initiating tendency of influencers. The model additionally embeds the aptitude of an influencer to create lengthy cascades in the norm of the embedding. That way, the norm of an influencer vector is used to estimate the expected influence spread and, thus, minimize the pool of potential seed candidates Then, following suit from recent model-independent algorithms [Lag+18; Vas+17b], we overlook the diffusion model and connect each candidate seed (influencer) and every susceptible node with a diffusion probability using the dot product of their respective embeddings, forming a bipartite network. Apart from removing the time-consuming simulations, diffusion probabilities have the advantage of capturing higher-order correlations that diffusion models fail to due to their Markovian nature. This allows to reformulate the network as a bipartite graph, introducing a greedy solution to influence maximization with theoretical guarantees. To evaluate the performance of the algorithm, the quality of the produced seed set is determined by a set of unseen cascades from future time steps, similar to a train and test split in machine learning. We consider this assessment strategy more reliable than traditional evaluations based on simulations as it relies on real traces of influence.

5.2.1 *Learning Influencer Vectors*

The first part of the methodology deals with INFECTOR, a multi-task learning model that captures simultaneously the influence between nodes and the aptitude of a node to create massive cascades.

NODE-CONTEXT EXTRACTION. We aim to design a model that learns representations suitable for scalable model-independent influence maximization. We start from the context creation process that produces the input to the network. In previous node-to-node influence learning models, initiating a cascade is considered equally important with participating in a cascade created by someone else [Fen+18]. Thus, the context of a node is derived by the nodes occurring after it in a cascade. This process requires the creation of the propagation network, meaning going through every node in the cascade and iterating over the subsequent nodes to search for a directed edge in the network—a time-consuming procedure. To overcome this challenge, we focus on the nature of influence maximization, where the ultimately selected seed users must exert influence over other nodes. Consequently, we can accelerate the process by leveraging the differences between influencers and simple users in terms of characteristics in sharing content. We argue that an influencer's strength lies in the cascades she initiates and evaluate this hypothesis through an exploratory analysis. To validate our hypothesis, we utilize the cascades of *Sina Weibo*, a large-scale social network accompanied by retweet cascades. The dataset is split into train and test cascades based on their time of occurrence, and each cascade represents a tweet and its set of retweets. We keep the 18K diffusion cascades from the last month of recording as a test set and the 97K from the previous 11 months as a train set. We rank all users that initiated a cascade in the test set based on three measures of success: the number of test cascades they spawn, their cumulative size, and the number of Distinct Nodes Influenced (DNI) [Du+13; PMM18; PMV18], which is the set of nodes that participated in these test cascades. We bin the users into three categories based on their success in each metric, and for each category, we compute the total cascades the users start in the training set as opposed to those they participate in.

Here, we examine the contrast between the behavior of the influencers and normal users, meaning how more probable it is for an influencer to start cascades compared to normal users. As we see in Fig. 5.1, users belonging to the top category of the test set are much more prone to create cascades than those belonging to the mid and low categories. Since our end goal is to find influencers for our algorithm, this observation allows us to focus solely on the initiators of the cascades rather than every node in the cascade. Moreover, we observe that influencers on *Sina Weibo* are more prone to create cascades than participating in them. This means that by overlooking their appearances inside the cascades of others, we do not lose too much information regarding their influence relationships, as most of them start the cascades. Thus, for our purpose, the context is extracted exclusively for the cascade initiator and will be comprised of all nodes that participate in it. Moreover, we will consider the copying time between the initiator and the re-poster, based on empirical observation on the effect of time in influence [GBL10]. That way, an influencer u's context will be created by sampling over all nodes v in a given cascade $\mathcal{C}_u$ that u started, with probability inversely proportional to their copying time. In this way, the faster v's retweet is, the more probable it will appear in the context of u, following this formula $P(v|\mathcal{C}_u) \sim \frac{(t_u - t_v)^{-1}}{\sum_{v' \in \mathcal{C}_u} (t_u - t'_v)^{-1}}$. As we will present shortly, this type of context allows the model to compute diffusion probabilities, i.e., influence between nodes with more than one hop distance in the network.

THE INFECTOR MODEL. The diffusion probabilities (DPs) are the basis for our model-independent approach and exhibit important practical advantages over influence probabilities, as we further analyze below. We use a multi-task neural network [Car97] to simultaneously learn an influencer's aptitude to create long cascades and the diffusion probabilities between her and the re-posters. We chose to extend the typical influence

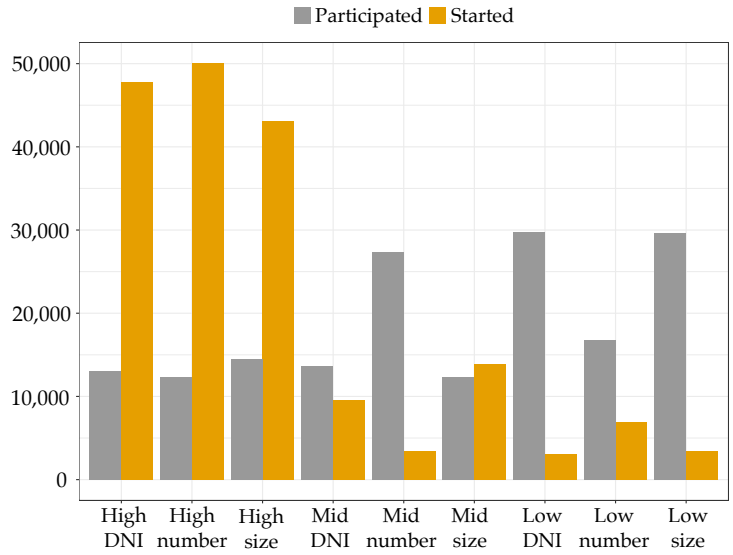


Figure 5.1: Influencer's initiating versus participating in diffusion cascades. 'DNI' stands for distinct nodes influenced, 'size' stands for cascade size, and 'number' stands for number of cascades started.

learning architecture in a multi-task learning setting because (i) the problem could naturally be broken into two tasks, and (ii) theoretical and applied literature suggests that training linked tasks together improves overall learning [EP04].

Given an input node u, the first task is to classify the nodes it will influence and the second to predict the size of the cascade it will create. Figure 5.2 shows an overview of the proposed INFECTOR model. There are two types of inputs. The first is the training set comprised of the node-context pairs. As defined in the previous paragraph, given a cascade t with length m, we get a set $\mathbf{X}^t = \{(\mathbf{x}^1, \mathbf{y}_t^1), (\mathbf{x}^1, \mathbf{y}_t^2), \ldots, (\mathbf{x}^m, \mathbf{y}_t^m)\}$, where $\mathbf{x} \in \mathbb{R}^I$ and $\mathbf{y}_t \in \mathbb{R}^N$ are one hot encoded nodes, with I the number of influencers in the train set and N the number of nodes in the network. The second input of the model is a similar set X^c, where instead of a vector, e.g., $\mathbf{y}_t^1$, there is a scalar y_c^1 denoting the length of that cascade, initiated by $\mathbf{x}^1$. To perform joint learning of both tasks, we mix the inputs following the natural order of the data; given a cascade, we first input the 'influencer-context' pairs extracted from it and then the 'influencer-cascade length' pair, as shown in Fig. 5.2. Here, $\mathbf{O} \in \mathbb{R}^{I \times E}$, with E being the embeddings size, represents the source embeddings, $\mathbf{O}_u \in \mathbb{R}^{1 \times E}$ the embeddings of cascade initiator u, $\mathbf{T} \in \mathbb{R}^{E \times N}$ the target embeddings, and $C \in \mathbb{R}^{E \times 1}$ is a constant vector initialized to one. Note that $\mathbf{O}_u$ is retrieved by multiplying the one-hot vector of u with the embedding matrix $\mathbf{O}$. The first output of the model represents the diffusion probability $p_{u,v}$ of the source node u for a node v in the network. It is created through a softmax function with a cross-entropy loss. The second output aims to regress the cascade length, which has undergone min-max normalization relative to the rest of the cascades in this set, and hence, a sigmoid function is used.

The main difference between INFECTOR and similar node-to-node influence learning methods is that it computes diffusion probabilities and does not require the underlying social network, in contrast to influence probabilities which are assigned to edges of the network [BLG16; Fen+18]. Intuitively, diffusion probability is the probability of the susceptible node appearing in a diffusion started by the influencer, independently of the two nodes' distance in the graph. This means that the underlying influence paths

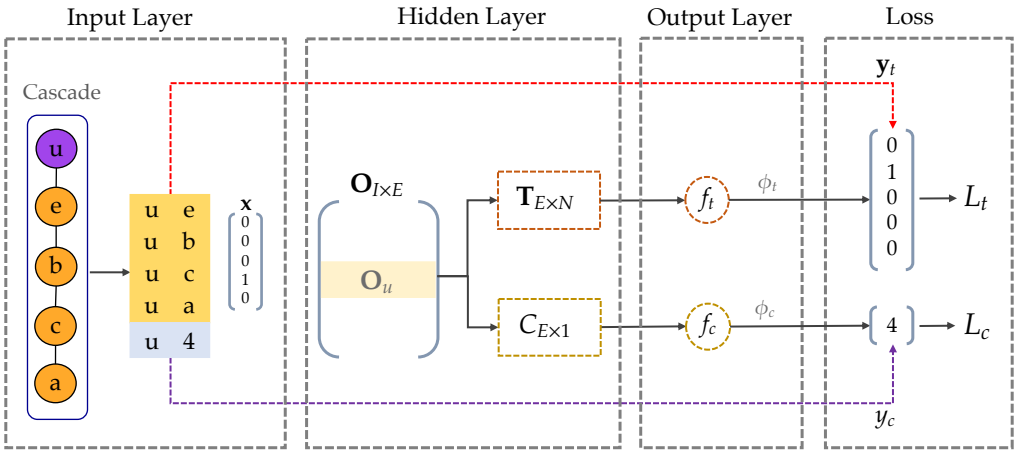


Figure 5.2: Schematic representation of the INFECTOR model. Given the cascade, a sequence of pairs is extracted, each pair consisting of the initiator node $\mathbf{x}$ (one-hot embedding), which is the *input*, and one of the "infected" nodes $\mathbf{y}_t$, i.e., e, b, etc. which is the *output*. The last pair is the initiator and the cascade size y_c, i.e., 4 as output. After the initiator's embedding lookup through the origin embeddings $\mathbf{O}$, the vector passes through $\mathbf{T}$ and f_t if the output is another node or through $\mathbf{C}$ and f_c if the output is scalar. The loss functions are log-loss L_t for node output or mean squared error L_c for scalar output.

from the seed to the infected node are included implicitly, which changes drastically the computation of influence spread. Please note that our method is ultimately geared towards influence maximization, which is why we emphasize the activity of influencers and overlook the rest of the nodes. If our objective was a purely predictive task, such as predicting the cascade size or the next infected node, the effectiveness of our learning mechanism would not be evident.

5.2.2 *Influence Maximization with Influencer Vectors*

In the second part of the methodology, we aim to perform fast and accurate influence maximization using the representations learned by INFECTOR and their properties. Initially, we will use the combination of the embeddings that form the diffusion probability of every directed pair of nodes. The diffusion probabilities derived from the network can define a matrix

$$\mathbf{D} = \begin{bmatrix} f_t(\mathbf{O}_1 T) \\ \vdots \\ f_t(\mathbf{O}_I T) \end{bmatrix}, \tag{5.1}$$

which consists of the nodes that initiate train cascades (influencers) in one dimension and all susceptible nodes in the other. Even though influencers are fewer than the total nodes, $\mathbf{D}$ can still be too memory-demanding for real-world datasets. To overcome this, we can keep the top P% influencers based on the norm of their influencer embedding $|\mathbf{O}_u|$ to reduce space, depending on the device. Recall that, the embeddings are trained such that their norm captures the influencers' potential to create lengthy cascades because $\mathbf{O}_u C = \sum_i^E \mathbf{O}_u(i) = |O|$ since C is constant. Subsequently, $\mathbf{D}$ can be interpreted as a bipartite network where the left side nodes are the candidate seeds for influence maximization, and each of them can influence every node on the right side where the

rest of the network resides. Since all edges are directed from left to right, no paths with lengths more than one exist. This means that the probability of an edge can only define the infection of one node, and it is independent of the infection of the rest—hence, we do not require a diffusion model to estimate the spread.

To formulate the influence spread, we will leverage the embedding of a candidate seed u to compute the fraction of nodes that is expected to influence:

$$\lambda_u = \left\lceil N \frac{||\mathbf{O}_u||_2}{\sum_{u' \in \mathcal{I}} ||\mathbf{O}_{u'}||_2} \right\rceil, \tag{5.2}$$

where $\mathcal{I}$ is the set of candidate seeds. The term resembles the norm of u relative to the rest of the influencers and the network size. It is basically computing the amount of the network u will influence. Since a seed s can influence a certain number of nodes, we can use the diffusion probabilities to identify the top λ_s nodes it connects to. Moreover, we must consider the diffusion probability values to avoid selecting nodes with big λ_s but an overall small probability of influencing nodes. Consequently, the influence spread is defined by the sum of the top λ_s diffusion probabilities as

$$\sigma'(s) = \sum_{j}^{\lambda_s} \hat{\mathbf{D}}_{s,j}, \tag{5.3}$$

where $\hat{\mathbf{D}}_s$ are the diffusion probabilities of seed s sorted in descending order. As mentioned above, once added to the seed set, the seed's influence set is considered infected and removed from $\mathbf{D}$. This means that a seed's spread will never get bigger in two subsequent rounds. Thus, in the proposed IMINFECTOR algorithm, we employ the CELF trick [Les+07] to accelerate our computation. CELF is an improved version of the GREEDY algorithm for influence maximization that exploits the property of submodularity to select seed nodes efficiently. By maintaining a sorted list of nodes based on their influence spread, CELF identifies the best node with the highest marginal gain in each iteration, resulting in significantly faster execution times without sacrificing effectiveness. We further prove that the influence spread function is monotonic and submodular, thus retaining the theoretical guarantees of the GREEDY algorithm by Kempe *et al.* [KKT03]. A detailed description of the IMINFECTOR algorithm is given in the corresponding article [PMV22].

5.2.3 *Experimental Evaluation of IMINFECTOR*

DATASETS. We have used three real-world graphs accompanied by ground-truth diffusion cascades. *Digg* is a social network with 280K nodes, 2.2M edges, and 3.5K cascades with an average cascade size of 847 [LG10]. *MAG* is a co-authorship network from Microsoft Academic Graph, having 1.4M nodes, 15.9M edges, and 181K cascades with an average size of 29 [Qiu+18a]. Finally, *Seina Weibo* is a social network (similar to X, formerly called Twitter) with 1.1M nodes, 225M edges, and 115K cascades with an average size of 148 [Zha+13].

BASELINE MODELS. Most traditional influence maximization algorithms, such as the GREEDY algorithm [KKT03] and CELF [Les+07], do not scale to the networks we have used here for evaluation. Thus, we employ the following scalable algorithms. K-CORE identifies influential nodes based on the k-core decomposition of a graph [Kit+10; MRV16; Mal+20].

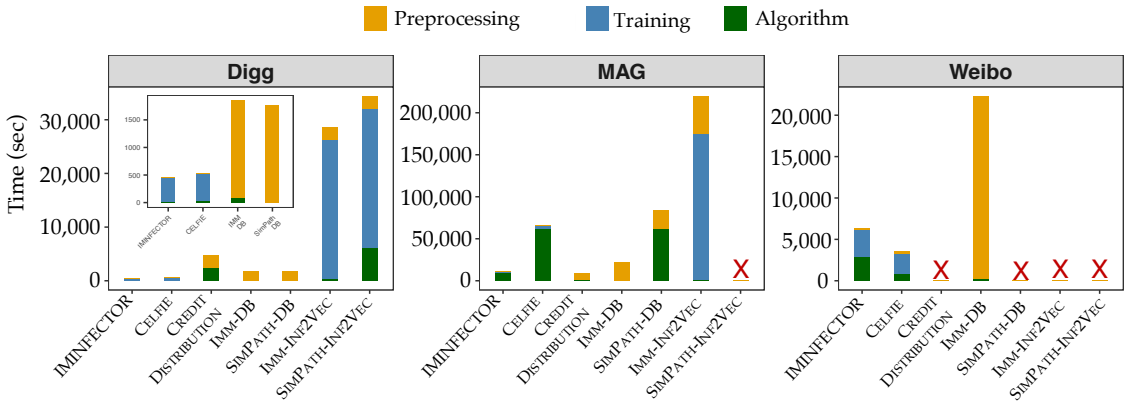


Figure 5.3: Time comparison of different models. Methods that could not scale to the dataset size are marked with X.

AVG-CASCADE-SIZE seeks the top nodes based on the average size of their cascades in the train set [Bak+11]. SIMPATH is a heuristic that capitalizes on the locality of influence pathways to reduce the cost of simulations in the influence spread [GLL11b]. IMM is a scalable algorithm based on reachable sets with theoretical guarantees regarding the influence spread [TSX15]. CREDIT-DISTRIBUTION uses cascade logs and the network edges to assign influence credits and derive a seed set [GBL11]. Finally, we use CELFIE [PMV20], a previous version of IMINFECTOR, which relies on the INF2VEC model [Fen+18] to perform influence learning.

Comparing graph-based algorithms such as SIMPATH and IMM to models that use both the graph structure and cascades is not a fair comparison, as the latter exploits more information. To make the comparison equitable, each graph-based method is coupled with two influence learning approaches, based on the diffusion cascades, that provide the influence maximization methods with influence weights on the edges. In the first one, denoted as DATA-BASED (DB), assuming that the 'follow' edge $u \rightarrow v$ exists in the network, the edge probability is set to the number of times node v has copied (e.g., retweeted) u, relative to the total activity (e.g., number of posts) of u [GBL10]. The second approach corresponds to INF2VEC, a shallow neural network performing influence learning based on the co-occurrences of nodes in diffusion cascades and the underlying network [Fen+18].

EXPERIMENTAL SETUP AND RESULTS. We split the datasets into train and test cascades based on their time of occurrence. The methods utilize the train cascades and/or the underlying network to define a seed set. The train cascades amount for the first 80% of the whole set, and the rest is left for testing. The evaluation is two-fold: computational time and seed set quality, similar to previous literature in influence maximization. We evaluate the quality of the predicted seed set using the *Distinct Nodes Influenced* (DNI) metric, which is the combined set of nodes that appear in the test cascades that are initiated from each one of the chosen seeds [PMM18; Du+13; PMV18]. Finally, since our datasets differ significantly in terms of size, we have to use different seed set sizes for each one. For *MAG*, which has 205,839 initiators in the train set, we test it on 10,000, *Weibo* with 26,158 is tested on 1,000, and *Digg* with 537 has a seed set size of 50.

Figure 5.3 shows the computational time of the examined methods, separated based on different parts of the model. Figure 5.4 depicts the estimated quality of each seed

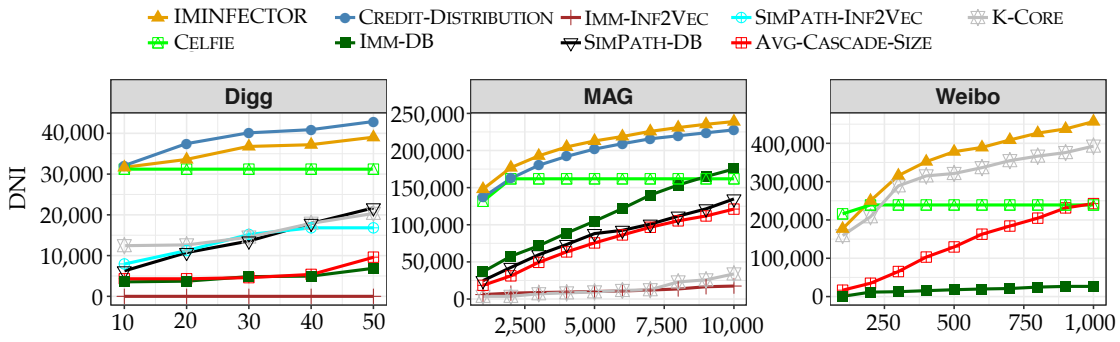


Figure 5.4: The quality of the seed set derived by each method in different sizes, measured by the seed set's number of Distinct Nodes Influenced (DNI) in the test cascades.

set. Regarding quality, we can see that IMINFECTOR surpasses the benchmarks in two datasets. In *Digg*, CREDIT-DISTRIBUTION performs better but is almost ten times slower because of the average cascade size in the dataset. All methods could scale in *MAG* except for SIMPATH with INF2VEC weights because, in contrast to DB, INF2VEC retains all the edges of the original *MAG* network, for which SIMPATH takes more than one week to run. IMM's performance is lower than expected, highlighting the difference between data-driven means of evaluation and the traditional simulations of diffusion models, which have been used in the literature due to the absence of empirical data.

In general, we see that IMINFECTOR provides a fair balance between computational efficiency and accuracy. Most importantly, it exhibits such performance using only the cascades, while the rest of the baselines use both the graph structure and cascades. Being unaffected by the network size, IMINFECTOR scales with the average cascade size and the number of cascades, making it suitable for real-world applications.

5.3 MAXIMIZING INFLUENCE WITH GRAPH NEURAL NETWORKS

> This section is based on material from a conference article co-authored with George Panagopoulos, Nikolaos Tziortziotis, and Michalis Vazirgiannis published in the *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)* [Pan+23].

In Sec. 5.2, we formulated an influence maximization method tailored for datasets with diffusion cascades. Nevertheless, many real-world graphs lack associated ground truth cascades. Consequently, this section introduces a neural approach to influence maximization based on diffusion models. In particular, we present GLIE, a GNN that provides efficient influence estimation for a given seed set and a graph with influence probabilities. It can be used as a standalone influence predictor with competitive results for graphs up to 10 times larger than the train set. We further leverage GLIE for influence maximization, combining it with CELF [Les+07], which typically does not scale beyond graphs with thousands of edges. The proposed method runs in graphs with millions of edges in seconds and exhibits better influence spread than a state-of-the-art algorithm and previous GNN-based methods for influence maximization. In addition, we propose PUN, a method that leverages GLIE's representations to compute the number of neighbors predicted to be uninfluenced, using it as an approximation to the marginal gain. We prove PUN's influence spread is submodular and monotone and hence can be optimized greedily

with theoretical guarantees, in contrast to most prior learning-based methods. The experimental evaluation indicates that PUN provides a good balance between influence quality and efficiency.

5.3.1 *Influence Estimation with GNNs*

Here, we discuss GLIE (*Graph Learning-based Influence Estimation*), a GNN model that aims to learn how to estimate the influence of seed set $\mathcal{S}$ over a graph $G = (\mathcal{V}, \mathcal{E})$. Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be the adjacency matrix and $\mathbf{X} \in \mathbb{R}^{n \times d}$ be the features of nodes, representing which nodes belong to the seed set by 1 and 0 otherwise:

$$\mathbf{X}_u = \begin{cases} \{1\}^d, & u \in \mathcal{S} \\ \{0\}^d, & u \notin \mathcal{S} \end{cases} . \tag{5.4}$$

For the analysis that follows, we set $d = 1$. More dimensions will become meaningful when we parameterize the problem. If we normalize $\mathbf{A}$ by each row, we form a row-stochastic transition matrix as

$$\mathbf{A}(u, v) = p_{vu} = \begin{cases} \dfrac{1}{\deg(u)}, & v \in \mathcal{N}(u) \\ 0, & v \notin \mathcal{N}(u) \end{cases} , \tag{5.5}$$

where $\deg(u)$ is the in-degree of node u and $\mathcal{N}(u)$ is the set of neighbors of u. Based on the weighted cascade model [KKT03], each row u stores the probability of node u being influenced by each of the other nodes that are connected to it by a directed link $v \to u$. The influence probability $p(u|\mathcal{S})$ resembles the probability of a node u getting influenced if its neighbors belong in the seed set, i.e., during the first step of the diffusion. We can use message passing to compute a well-known upper bound $\hat{p}(u|\mathcal{S})$ of $p(u|\mathcal{S})$ for u:

$$\hat{p}(u|\mathcal{S}) = \mathbf{A}_u \cdot \mathbf{X} = \sum_{v \in \mathcal{N}(u) \cap \mathcal{S}} \frac{1}{\deg(u)} = \tag{5.6}$$

$$\sum_{v \in \mathcal{N}(u) \cap \mathcal{S}} p_{vu} \geq 1 - \prod_{v \in \mathcal{N}(u) \cap \mathcal{S}} (1 - p_{vu}) = p(u|\mathcal{S}), \tag{5.7}$$

where the second equality stems from the definition of the weighted cascade and the inequality from the proof in [Zho+15]. As the diffusion covers more than one hop, the derivation requires repeating the multiplication to approximate the total influence spread. To be specific, computing the influence probability of nodes not adjacent to the seed set requires estimating the probability of their neighbors being influenced by the seeds recursively. If we let $\mathbf{H}_1 = \mathbf{A} \cdot \mathbf{X}$, and we assume the new seed set $\mathcal{S}^t$ to be the nodes influenced in the step $t - 1$, their probabilities are stored in $\mathbf{H}_t$, much like a diffusion in discrete time. We can then recompute the new influence probabilities with $\mathbf{H}_{t+1} = \mathbf{A} \cdot \mathbf{H}_t$.

Theorem 5.1. *The repeated product $\mathbf{H}_{t+1} = \mathbf{A} \cdot \mathbf{H}_t$ computes an upper bound to the real influence probabilities of each infected node at step $t + 1$.*

In reality, due to the existence of cycles, two problems arise. Firstly, if the process is repeated, the influence of the original seeds may increase, which contrasts with the independent cascade model. This can be controlled by minimizing the repetitions, e.g., four repetitions cause the original seeds to be able to reinfect other nodes in a network with triangles. To this end, we leverage up to three GNN layers. Another

problem due to cycles pertains to the probability of neighbors influencing each other. In this case, the product of the complementary probabilities in (5.7) does not factorize for the non-independent neighbors. This effect was analyzed extensively in [LS19], showing that the influence probability computed by $p(u|\mathcal{S})$ is an upper bound on the real influence probability for graphs with cycles. Intuitively, the product that represents non-independent probabilities is larger than the product of independent ones. This renders the real influence probability, complementary to the product, smaller than what we compute.

From the formulation above, we can contend that the estimation $\hat{p}(u|\mathcal{S})$ provides an upper bound on the real influence probability—and we can use it to compute an upper bound to the real influence spread of a given seed set, i.e., the total number of nodes influenced by the diffusion. Since message passing can inherently compute an approximation of influence estimation, we can parameterize it to learn a function that tightens this approximation based on supervision. In our neural network architecture, each layer consists of a GNN. Starting from $\mathbf{H}_0 = \mathbf{X} \in \mathbb{R}^{n \times d}$, we have:

$$\mathbf{H}_{t+1} = \text{ReLU}([\mathbf{H}_t, \mathbf{A}\mathbf{H}_t]\mathbf{W}_t). \tag{5.8}$$

The readout function that summarizes the graph representation based on all nodes' representations is a summation with skip connections:

$$\mathbf{H}_{\mathcal{S}}^{G} = \sum_{v \in \mathcal{V}} [\mathbf{H}_0^v, \mathbf{H}_1^v, \ldots, \mathbf{H}_t^v]. \tag{5.9}$$

This representation captures the probability of all nodes being active throughout each layer. The output that represents the predicted influence spread is derived by:

$$\hat{\sigma}(\mathcal{S}) = \text{ReLU}(\mathbf{H}_{\mathcal{S}}^{G}\mathbf{W}_o). \tag{5.10}$$

The loss function used here is a simple least squares regression. Note that, in the case where $\mathbf{W}_t$ is an untrained positive semidefinite Gaussian random matrix in $[0, 1]$, the representations of each layer $\mathbf{H}_t^v$ would correspond to the upper bound of the influence probability of seed set's t-hop neighbors [LS19]. This upper bound is not retained once the weights $\mathbf{W}_t$ are trained. In our approach, the parameters of the intermediate layers $\mathbf{W}_t$ are trained such that the upper bound is reduced and the final layer $\mathbf{W}_o$ can combine the probabilities to derive a cumulative estimate for the total number of influenced nodes. We empirically verify this by examining the layer activations seen in Fig. 5.5. The heatmaps indicate a difference between columns (nodes) expected to be influenced, meaning we could potentially predict not only the number but also who will be influenced. However, since $\hat{\sigma}$ is derived by multiple layers, the relationships and thresholds to determine the exact influenced set are not straightforward.

5.3.2 *Influence Maximization with Glie*

FIRST APPROACH: COMBINE GLIE WITH CELF. Similar to the IMINFECTOR model presented in the previous section, we can leverage the Cost Effective Lazy Forward (CELF) algorithm [Les+07] for influence maximization. Thus, as a first approach, we propose an adaptation where we substitute the original CELF's influence estimation component (based on Monte Carlo simulations) with the output of GLIE. Since we do not prove the submodularity of $\hat{\sigma}$, we can not contend that the theoretical guarantee is retained, so we use this as a heuristic. CELF-GLIE has two main computational bottlenecks. First,

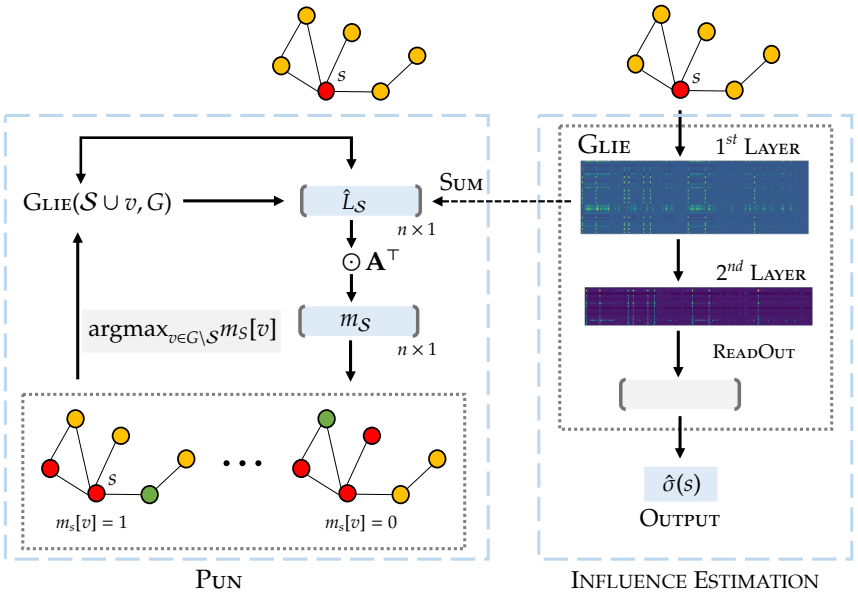


Figure 5.5: A schematic representation of the pipeline. The layers of GLIE are depicted by a heatmap of an actual seed during inference time, showing how the values vary through different nodes (columns).

it requires computing the initial influence estimation for every node in the first step. Second, although it alleviates the need to test every node in every step, it still requires performing influence estimation for at least one node in each step. We aim to alleviate both limitations in the next paragraph.

SECOND APPROACH: POTENTIALLY UNINFLUENCED NEIGHBORS (PUN). Computing the influence spread of every node in the first step is computationally demanding. We thus seek a method that can surpass this obstacle and provide adequate performance. We first utilize the activations mentioned above to define the set of influenced nodes on the step that corresponds to that layer. Let $\hat{L}_S, L'_S \in \{0,1\}^n$ be the binary vectors with 1's in nodes predicted to be uninfluenced and nodes predicted to be influenced, respectively:

$$\hat{L}_S = \mathbb{1}\left\{\sum_{i=0}^{d_1} \mathbf{H}_1^i \leq 0\right\} \qquad L'_S = \mathbb{1}\left\{\sum_{i=0}^{d_1} \mathbf{H}_1^i > 0\right\}. \tag{5.11}$$

This vector contains a label for each node whose sign indicates if it is predicted to be influenced. L'_S provides a rough estimate, but it allows for a simpler influence spread, which we can optimize greedily:

$$\sigma^m(S) = |L'_S|. \tag{5.12}$$

We can use $\hat{L}_S$ and message passing to predict the amount of a node's neighborhood that remains uninfluenced, i.e., the *Potentially Uninfluenced Neighbors* (PUN), weighted by the respective probability of influence for a node u,

$$m_S[u] = \sum_{v \in \mathcal{N}(u)} \mathbf{A}(u,v)\hat{L}_v = \mathbf{A}_u^\top \cdot \hat{L}_S \in \mathbb{R}^{n\times 1}. \tag{5.13}$$

For efficiency, we can compute $m_{\mathcal{S}} = \mathbf{A}^\top \hat{L}$, which can be considered an approximation to all nodes' marginal gain on their immediate neighbors. We can thus optimize this using argmax$(m_{\mathcal{S}})$, as shown in Fig. 5.5. To establish that σ^m can be optimized greedily with theoretical guarantees, we prove its monotonicity and submodularity.

Theorem 5.2. *The influence spread σ^m is monotone and submodular.*

PUN can be seen in the left part of Fig. 5.5. We start by setting the first seed as the node with the highest degree, which can be considered a safe assumption as, in practice, it is always part of the seed set. We use GLIE$(\mathcal{S}, G)$ to retrieve $\hat{L}_{\mathcal{S}}$, which we leverage to find the next node based on argmax$_{v \in G \setminus \mathcal{S}} m_{\mathcal{S}}[v]$ and the new $\hat{L}_{\mathcal{S} \cup \{v\}}$. One disadvantage of PUN is that σ^m is an underestimation of the predicted influence. Contrasted with the upper bound of the DMP (Direct Message Passing) model [LS19], σ^m is not as accurate as $\hat{\sigma}$, but allows us to compute efficiently a submodular proxy for the marginal gain.

5.3.3 *Experimental Evaluation of Glie*

Here, we provide an overview of the experimental analysis of GLIE on the tasks of influence learning and maximization. An elaborate list of experiments is presented in the corresponding article [Pan+23].

INFLUENCE LEARNING. To train for the influence estimation task, we create a set of labeled samples, each consisting of the seed set $\mathcal{S}$ and the corresponding influence spread $\sigma(\mathcal{S})$. We have generated 100 Barabási-Albert [BA99] and Holme-Kim [HK02] undirected graphs ranging from 100 to 500 nodes. 60% are used for training, 20% for validation, and 20% for testing. The influence probabilities are assigned based on the weighted cascade model, i.e., a node u has an equal probability $1/\deg(u)$ to be influenced by each of her $\mathcal{N}(u)$ nodes. To label the samples, we run the CELF algorithm using 1,000 Monte Carlo (MC) simulations for up to 5 seeds. The optimum seed set for sizes 1 to 5 is stored, along with 30 random negative samples for each seed set size. Each training sample for GLIE corresponds to a triple of a graph G, a seed set $\mathcal{S}$, and a ground truth influence spread $\sigma(\mathcal{S})$ that serves as a label to regress on. The random seed sets are used to capture the average influence spread expected for a seed set of about that size. This creates "average samples" which would constitute the whole dataset in other problems. In influence maximization, however, the difference in σ between an average seed set and the optimal can be significant. Hence, training solely on the random sets would render our model unable to predict larger values corresponding to the optimum. That is why we added the samples of the optimum seed set computed using CELF. Besides the small random graphs presented above (denoted as *Test* in the experiments), we have evaluated the model on larger power-law graphs (*Large*; 1K – 2K nodes) as well as on three real-world graphs, namely *Crime* (1K nodes, 3K edges), a biological network *HI-II-14* (4K nodes, 16K edges), and a collaboration network *GR Colab* (5K nodes, 29K edges).

The real graphs are evaluated for varying seed set sizes, from 2 to 10, to test our model's capacity to extrapolate to larger seed set sizes. We have compared the accuracy of influence estimation to DMP [LS19]. The average error throughout all datasets and the average influence can be seen in Table 5.1, along with the average time.

INFLUENCE MAXIMIZATION. We have considered several influence maximization baseline models, both algorithms with theoretical guarantees and heuristics. IMM [TSX15]

Graph (seeds)	DMP		GLIE	
	MAE	**Time**	**MAE**	**Time**
Test (1 – 5)	0.076	0.05	**0.046**	**0.0042**
Large (1 – 5)	**0.086**	0.44	0.102	**0.0034**
Crime (1 – 10)	**0.009**	0.11	0.044	**0.0029**
HI-II-14 (1 – 10)	**0.041**	2.84	0.056	**0.0034**
GR Colab (1 – 10)	0.122	4.32	**0.084**	**0.0042**

Table 5.1: Average MAE divided by the average influence and time (in seconds) throughout all seed set sizes and samples, along with the real average influence spread.

Graph	GLIE-CELF	PUN	K-CORE	PMIA	DEGDISC	IMM	DEEPIS-CELF	FINDER
Crime	661	657	647	656	644	650	501.61	642
GR Colab	1,617	**1,626**	701	1,566	1415	835.40	1,617	1,286
HI-II-14	2,685	**2,688**	2,540	2,685	2,614	2,668	1602.5	2,625
Enron	17,601	**17,614**	13,015	17,534	16,500	17,497	-	17,244
Facebook	10,981	10,626	6,434	7,688	10,309	**11,007**	-	10,801
Youtube	246,439	244,579	110,409	242,057	236,726	**247,178**	-	50,435

Table 5.2: Influence spread computed by 10,000 simulations of the Independent Cascade model for 200 seeds.

is based on sketches to approximate the influence spreading, achieving remarkable performance. FINDER [Fan+20] is a reinforcement learning model where the reward is based on the size of the giant connected component. In particular, each new node (seed) chosen aims to dismantle the network as much as possible. PMIA [WCW12] computes the influence spread based on local approximations. DEGDISC (Degree Discount) [CWY09] builds a seed set using the node's degree, which is recomputed based on the current seed set and its influence. Finally, the K-CORE [Kit+10] model identifies influential spreaders based on the k-core decomposition of the graph. Regarding the datasets, besides the ones used above for influence learning, we also consider three real-world social graphs, namely *Enron* (34K nodes, 362K edge), FACEBOOK (63K nodes, 1.63M edges), and YOUTUBE (1.13M nodes, 5.97M edges).

The results for the influence spread of 200 seeds as computed by simulations of the independent cascade model can be seen in Table 5.2, while the time results are shown in Tables 5.3 and 5.4 (to have a fair evaluation, we compare algorithms with theoretical guarantees separately from heuristics). One can see that GLIE-CELF exhibits overall superior influence quality compared to the rest of the methods but is quite slower. PUN requires only one influence estimation in every step and no initial computation. It exhibits 3 to 60 times acceleration compared to IMM while its computational overhead moving from smaller to larger graphs is sublinear to the number of nodes. Regarding influence quality, PUN is first or second in most datasets, and this effect becomes clearer as the seed set size increases. DEGDISC is faster than PUN in smaller graphs but slower in larger and overall worse in seed set quality. PMIA provides medium seed set quality but is computationally inefficient. IMM is not the fastest method, but it is very accurate, especially for smaller seed set sizes. FINDER exhibits the least accurate performance, which is understandable given that it solves a relevant connectivity problem and not directly influence maximization. Overall, we can contend that PUN provides the best accuracy-efficiency trade-off from the examined methods.

Graph	GLIE-CELF	PUN	IMM	FINDER
Crime	2.00	0.25	**0.19**	0.41
Gr Colab	4.55	**0.26**	0.95	2.36
HI-II-14	2.19	**0.27**	1.29	1.01
Enron	15.49	**0.97**	10.47	9.30
Facebook	287.70	**3.10**	171.25	56.80
Youtube	151.33	**28.92**	82.13	191.00

Table 5.3: Computational time in seconds (vs. algorithms).

PUN	DEGDISC	K-CORE	PMIA
0.25	0.21	0.06	**0.04**
0.26	0.80	**0.13**	1.50
0.27	1.36	0.14	**0.12**
0.97	26.74	2.06	2.17
3.10	22.77	9.29	10.62
28.92	4006.29	54.38	74.91

Table 5.4: Computational time in seconds (vs. heuristics).

5.4 DISCUSSION

In this chapter, we examined how graph representation learning can address the problems of influence learning and maximization in complex networks. We first introduced IMINFECTOR, a model-independent method to perform influence maximization using representations learned from diffusion cascades. A basic limitation of the approach is related to the assumption that the optimum seed set is comprised solely of nodes that initiate cascades. Besides, due to the absence of multiple datasets with diffusion cascades, our evaluation was limited to three datasets of varying sizes and characteristics. Then, we discussed GLIE, a GNN-based model for influence estimation, and the derived models for influence maximization. A practical advantage of a neural approach is the easy incorporation of complementary information, such as topic or user's characteristics [BBM12]. We still need to examine GLIE's behavior with such contextual information.

6

PERSPECTIVES AND FUTURE RESEARCH

This HDR manuscript has presented an overview of research activities at the intersection of machine learning and network science. The focus was on graph representation learning, examining various methodologies and concrete application domains. Chapter 2 presented random walk node embedding algorithms, emphasizing expressiveness and scalability. We first examined how the clustering structure of the graph can enhance node embeddings (Sec. 2.2). Then, we studied flexible embedding models, either by generalizing the modeling assumptions about node co-occurrences in random walks (Sec. 2.3) or by leveraging kernel methods (Sec. 2.4). Lastly, this chapter proposed a scalable embedding algorithm based on hashing techniques (Sec. 2.5).

Chapter 3 focused on representation learning for multilayer graphs, with particular emphasis on prediction tasks arising in the domain of computational biology. We first studied simple instances of random walk node embedding techniques extended to multilayer graphs (Sec. 3.2). Then, we discussed a joint matrix factorization framework for learning node representations on multilayer graphs (Sec. 3.3). These graph machine learning models can further facilitate data integration tasks when dealing with multiple input data sources. Finally, the last part of this chapter introduced a supervised learning methodology for link prediction in heterogeneous graphs, combining random walks with matrix factorization (Sec. 3.4). Specifically, we addressed this problem in the context of predicting missing drug-target interactions in data-driven drug discovery.

Going further, Chapter 4 explored aspects related to Graph Neural Networks. First, we studied the trade-off between over-smoothing and over-squashing, proposing a curvature-based algorithm to alleviate both phenomena (Sec. 4.2). Then, we introduced a hierarchical motif-based pooling operator for GNNs used to compute graph-level representations (Sec. 4.3). The techniques presented in these sections are theoretically grounded upon spectral graph theory and network science tools. As a last topic, we studied the design of explanation models for GNNs (Sec. 4.4).

Finally, Chapter 5 examined applications of graph representation learning in identifying influential spreaders in complex networks. We first focused on how real diffusion cascades can be leveraged to learn informative node representations that can later be used to perform influence maximization (Sec. 5.1). Then, we presented a methodology that relies on GNNs for efficient influence estimation (Sec. 5.3).

In the following paragraphs, we will conclude this manuscript by discussing ongoing and future research directions in the broader field of graph machine learning and network analysis.

GRAPH SELF-SUPERVISED LEARNING. In the absence of labeled data, GNN models for self-supervised learning constitute a promising new direction, with graph contrastive learning being of particular interest [Xie+23]. The problem is typically expressed as a two-step process. First, different graph views are generated via a (handcrafted) data augmentation process covering both topology and feature transformations. Then, a GNN model is used to learn representations of the different views. The representations are then optimized based on a contrastive loss function that pulls together views of positive samples (e.g., the same node across different views) while pushing apart negative pairs.

Despite the recent progress, most of the proposed models rely on formulations introduced in computer vision and natural language processing—and often do not fully leverage crucial graph semantics that could lead to better representations. For instance, observing an edge between two nodes in a social graph might be due to multiple latent factors (e.g., individuals sharing the same profession, living in the same city, or studying at the same university). Disentangling these factors within an augmentation strategy could lead to representations with improved predictive and generalization capabilities. Moreover, performing a systematic theoretical analysis toward understanding the limitations of graph self-supervised contrastive loss functions and their impact on generalization to new datasets and tasks is an important research direction to explore. Lastly, developing structure-aware loss functions that directly contrast subgraphs presents an intriguing direction that diverges from existing approaches. Such a model could be framed around recent advances in Optimal Transport, including the Fused Gromov-Wasserstein distance [Vay+19; BM+22].

SPATIOTEMPORAL GRAPH LEARNING. Several real-world prediction problems involve spatiotemporal data. Consider, for instance, sensors spread across different geographical areas measuring environmental conditions (e.g., temperature, pollution) or functional magnetic resonance imaging (fMRI) data measuring brain activity. Both settings produce data with inherently rich spatiotemporal structure, which can benefit from the relational inductive bias of graph-based modeling [Jin+23]. Indeed, considering the structure of the data, the pairwise relationships between time series can be represented by a graph, enabling GNNs and other types of graph learning models to be used. In our recent work, we have introduced a model that allows us to perform time series imputation with GNNs [CC+23]. A key challenge here is incorporating temporal and relational smoothness assumptions in the model. Besides, in most cases, the underlying graph structure is unknown. Although most approaches construct simple k-Nearest Neighbor (k-NN) or other correlation graphs, learning the graph structure along with the model parameters remains a challenging problem [Zho+23]. Lastly, enhancing such models with causal properties to capture causal influence effects among entities constitutes another interesting direction [Lim+15].

GRAPH MACHINE LEARNING FOR COMBINATORIAL OPTIMIZATION. Solving combinatorial optimization problems with machine learning is an emerging area of study [QCa+23]. In this manuscript, we have examined two such problems, namely graph clustering (Sec. 4.3) and influence maximization (Sec. 5.3). Apart from the scale of the graphs considered here, several other challenges must be addressed. First, the algorithms should capture and reflect the inherent structure of real graphs in the representations. This was, for instance, the case while designing a GNN model to capture the influence of nodes within its hidden representations in Chapter 5. Furthermore, the models should generalize to instances of varying sizes outside the training set (e.g., learn cluster assignments in graph partitioning for the test graph instances). While progress has been made, further efforts are needed to develop models balancing efficiency, generalization, and graph structure expressiveness.

GEOMETRIC GNNS. In many practical applications in molecular and chemical systems, the nodes of the graph have associated geometric attributes (e.g., coordinates) related to their position in the 3D space. In this context, geometric graphs represent the interaction of atoms (i.e., nodes) in the 3D space, encapsulating a range of physical symmetries such

as rotations and translations [Zha+23]. Existing GNN models often overlook this aspect, making them unsuitable for prediction tasks on geometric graphs. Recently, Geometric GNN architectures tailored to respect physical symmetries have emerged as flexible models of atomic systems such as small molecules, proteins, and materials. Nevertheless, such models often have to balance expressiveness and computational efficiency due to the requirement to learn invariant or equivariant functions. Our recent work moves towards this direction, proposing a model where symmetries are preserved through appropriate data augmentations rather than architectural constraints [Duv+23b]. For an in-depth exploration of this emerging field and its various challenges, we direct the reader to our recent survey article [Duv+23a]. Overall, we deem geometric GNNs and geometric deep learning, in general, a key tool to facilitate scientific discoveries and accelerate research [Wan+23].

GRAPH GENERATIVE MODELS. Generating realistic graphs is a longstanding problem in the graph mining [CF06], network science [LFR08], and machine learning [GZ23] communities. More recently, the progress in graph representation learning and the advances in generative models in computer vision and natural language processing paved the way for a new class of deep graph generative models. Several such models have been proposed, mainly relying on variational autoencoders, generative adversarial networks (GANs), normalizing flows, and diffusion models. One of the main challenges here stands from the discrete nature of graphs. Consider, for instance, the case of diffusion models in which a noise model progressively corrupts the data (forward process) while a denoising model strives to invert the process (reverse process) [Liu+23a]. However, representing the graph in a continuous space where Gaussian noise is added in the forward process could result in missing the structural properties of the graph [Vig+23]. Other challenges here include multimodal graph generation (e.g., considering textual and visual features) as well as time-aware graph generation focusing on dynamic graphs. Overall, the progress in graph generation can be instrumental in a plethora of practical applications, especially in molecule generation, including drug discovery and materials science.

A

LIST OF PUBLICATIONS

🏛: supervised Ph.D. or intern student since 2016

Journal Articles

J17. Jhon A. Castro-Correa, Jhony H. Giraldo, Mohsen Badiey, and Fragkiskos D. Malliaros. Gegenbauer Graph Neural Networks for Time-varying Signal Reconstruction. *IEEE Transactions on Neural Networks and Learning Systems (**TNNLS**)*, 2024. (Impact Factor: 10.45).

J16. Aref Einizade, Jhony H. Giraldo, Fragkiskos D. Malliaros, Sepideh Hajipour Sardouie. Estimation of a Causal Directed Acyclic Graph Process using Non-Gaussianity. ***Digital Signal Processing***, Elsevier, 2024. (Impact Factor: 2.9).

J15. Surabhi Jagtap🏛, Abdulkadir Çelikkanat🏛, Aurélie Pirayre, Frederique Bidard, Laurent Duval, and Fragkiskos D. Malliaros. BraneMF: Integration of Biological Networks for Functional Analysis of Proteins. ***Bioinformatics***, Oxford University Press, 2022. (Impact Factor: 6.931.)

J14. Surabhi Jagtap🏛, Aurélie Pirayre, Frederique Bidard, Laurent Duval, and Fragkiskos D. Malliaros. BRANEnet: Embedding Multilayer Networks for Omics Data Integration. ***BMC Bioinformatics***, 2022. (Impact Factor: 4.341).

J13. Bin Liu, Dimitrios Papadopoulos, Fragkiskos D. Malliaros, Grigorios Tsoumakas, Apostolos N. Papadopoulos. Multiple Similarity Drug-Target Interaction Prediction with Random Walks and Matrix Factorization. ***Briefings in Bioinformatics***, Oxford University Press, 2022. (Impact Factor: 13.994).

J12. Abdulkadir Çelikkanat🏛, Yanning Shen, and Fragkiskos D. Malliaros. Multiple Kernel Representation Learning on Networks. *IEEE Transactions on Knowledge and Data Engineering (**TKDE**)*, 2022. (Impact Factor: 6.97).

J11. Kavya Gupta🏛, Fateh Kaakai, Beatrice Pesquet-Popescu, Jean-Christophe Pesquet, and Fragkiskos D. Malliaros. Multivariate Lipschitz Analysis of the Stability of Neural Networks. ***Frontiers in Signal Processing***, 2022.

J10. Abdulkadir Çelikkanat🏛 and Fragkiskos D. Malliaros. Topic-Aware Latent Models for Representation Learning on Networks. ***Pattern Recognition Letters***, Elsevier, 2021. (Impact Factor: 3.255).

J9. George Panagopoulos🏛, Fragkiskos D. Malliaros, and Michalis Vazirgiannis. Multi-task Learning for Influence Estimation and Maximization. *IEEE Transactions on Knowledge and Data Engineering (**TKDE**)*, 2020. (Impact Factor: 4.935).

J8. Chloé Adam, Antoine Aliotti, Fragkiskos D. Malliaros, and Paul-Henry Cournède. Dynamic Monitoring of Software Use with Recurrent Neural Networks. *Data & Knowledge Engineering (**DKE**)*, Elsevier, 2020. (Impact Factor: 1.583).

J7. Fragkiskos D. Malliaros, Christos Giatsidis, Apostolos N. Papadopoulos, and Michalis Vazirgiannis. The Core Decomposition of Networks: Theory, Algorithms and Ap-

plications. *International Journal on Very Large Data Bases (**The VLDB Journal**)*, 2019. (Impact Factor: 1.973).

J6. Maria-Evgenia G. Rossi, Bowen Shi, Nikolaos Tziortziotis, Fragkiskos D. Malliaros, Christos Giatsidis, and Michalis Vazirgiannis. MATI: An Efficient Algorithm for Influence Maximization in Social Networks. ***PLOS One***, 13(11): e0206318, 2018. (Impact Factor: 2.776).

J5. Jordi Casas-Roma, Julián Salas, Fragkiskos D. Malliaros, and Michalis Vazirgiannis. *k*-Degree Anonymity on Directed Networks. *Knowledge and Information Systems (**KAIS**)*, Springer, 2018. (Impact Factor: 2.936).

J4. Fragkiskos D. Malliaros, Maria-Evgenia G. Rossi, and Michalis Vazirgiannis. Locating Influential Nodes in Complex Networks. ***Scientific Reports***, 6(19307) Nature Publishing Group, 2016. (Impact Factor: 3.998).

J3. Fragkiskos D. Malliaros and Michalis Vazirgiannis. Vulnerability Assessment in Social Networks under Cascade-Based Node Departures. *Europhysics Letters (**EPL**)*, 110(6): 68006, IOP Science, 2015. (Impact Factor: 2.095).

J2. Fragkiskos D. Malliaros, Vasileios Megalooikonomou, and Christos Faloutsos. Estimating Robustness in Large Social Graphs. *Knowledge and Information Systems (**KAIS**)*, 45(3): 645–678, Springer, 2015. (Impact Factor: 2.936).

J1. Fragkiskos D. Malliaros and Michalis Vazirgiannis. Clustering and Community Detection in Directed Networks: A Survey. ***Physics Reports***, 533(4): 95–142, Elsevier, 2013. (Impact Factor: 25.798).

Conference Papers

C37. George Panagopoulos🏛, Nikolaos Tziortziotis, Michalis Vazirgiannis, and Fragkiskos D. Malliaros. Maximizing Influence with Graph Neural Networks. In Proceedings of the *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (**ASONAM**)*, Marrakesh, Morocco, 2023.

C36. Jhony H. Giraldo, Konstantinos Skianis, Thierry Bouwmans, and Fragkiskos D. Malliaros. On the Trade-off between Over-smoothing and Over-squashing in Deep Graph Neural Networks. In Proceedings of the *31st ACM International Conference on Information and Knowledge Management (**CIKM**)*, Birmingham, UK, 2023. (Full Paper).

C35. Alexandre Duval🏛, Victor Schmidt, Alex Hernandez Garcia, Santiago Miret, Fragkiskos D. Malliaros, Yoshua Bengio, and David Rolnick. FAENet: Frame Averaging Equivariant GNN for Materials Modeling. In Proceedings of the *International Conference on Machine Learning (**ICML**)*, Hawaii, 2023.

C34. Jhon A. Castro-Correa, Jhony H Giraldo, Anindya Mondal, Mohsen Badiey, Thierry Bouwmans, and Fragkiskos D. Malliaros. Time-varying Signals Recovery via Graph Neural Networks. In Proceedings of the *IEEE International Conference on Acoustics, Speech, and Signal Processing (**ICASSP**)*, Rhodes Island, Greece, 2023.

C33. Jhony H. Giraldo, Sajid Javed, Arif Mahmood, Fragkiskos D. Malliaros, and Thierry Bouwmans. Higher-order Sparse Convolutions in Graph Neural Networks. In Proceedings of the *IEEE International Conference on Acoustics, Speech, and Signal Processing (**ICASSP**)*, Rhodes Island, Greece, 2023.

C32. Abdulkadir Çelikkanat, Fragkiskos D. Malliaros, and Apostolos N. Papadopoulos. NodeSig: Binary Node Embeddings via Random Walk Diffusion. In Proceedings of the *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (**ASONAM**)*, Istanbul, Turkey, 2022.

C31. Alexandre Duval and Fragkiskos D. Malliaros. Higher-order Clustering and Pooling for Graph Neural Networks. In Proceedings of the *31st ACM International Conference on Information and Knowledge Management (**CIKM**)*, Atlanta, Georgia, 2022. (Full Paper).
Also, presented at the *ICML Workshop on Topology, Algebra, and Geometry in Machine Learning (**TAG:ML**)*, Baltimore, Maryland, 2022.

C30. Surabhi Jagtap, Aurélie Pirayre, Frederique Bidard, Laurent Duval, and Fragkiskos D. Malliaros. BRANET: Graph-based Integration of Multi-omics Data with Biological a priori for Regulatory Network Inference. In Proceeding of the *International Conference on Computational Intelligence Methods for Bioinformatics and Biostatistics (**CIBB**)*, Virtual, 2021.

C29. Alexandre Duval and Fragkiskos D. Malliaros. GraphSVX: Shapley Value Explanations for Graph Neural Networks. *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (**ECML PKDD**)*, Virtual, 2021.

C28. Surabhi Jagtap, Abdulkadir Çelikkanat, Aurélie Pirayre, Frederique Bidard, Laurent Duval, and Fragkiskos D. Malliaros. Multiomics Data Integration for Gene Regulatory Network Inference with Exponential Family Embeddings. In Proceeding of the *European Signal Processing Conference (**EUSIPCO**)*, Virtual, 2021.

C27. Kavya Gupta, Beatrice Pesquet-Popescu, Fateh Kaakai, Jean-Christophe Pesquet, and Fragkiskos D. Malliaros. An Adversarial Attacker for Neural Networks in Regression Problems. *IJCAI AI Safety workshop (**AISafety**)*, Virtual, 2021.
Shortlisted for the best paper award.

C26. Konstantinos Skianis, Fragkiskos D. Malliaros, Nikolaos Tziortziotis, and Michalis Vazirgiannis. Boosting Tricks for Word Mover's Distance. In Proceeding of the *International Conference on Artificial Neural Networks (**ICANN**)*, Bratislava, Slovakia, 2020.

C25. Abdulkadir Çelikkanat and Fragkiskos D. Malliaros. Exponential Family Graph Embeddings. In Proceedings of the *AAAI Conference on Artificial Intelligence (**AAAI**)*, New York City, New York, 2020.

C24. George Panagopoulos, Michalis Vazirgiannis, and Fragkiskos D. Malliaros. Influence Maximization with Influence and Susceptibility Embeddings. In Proceedings of the *International AAAI Conference on Web and Social Media (**ICWSM**)*, Atlanta, Georgia, 2020.
Best paper honorable mention award.

C23. Abdulkadir Çelikkanat and Fragkiskos D. Malliaros. Learning Node Embeddings with Exponential Family Distributions. In Proceedings of the *NeurIPS Graph Representation Learning Workshop (**NeurIPS-GRL**)*, Vancouver, Canada, 2019.

C22. Abdulkadir Çelikkanat and Fragkiskos D. Malliaros. Kernel Node Embeddings. In Proceedings of the *7th IEEE Global Conference on Signal and Information Processing (**GlobalSIP**)*, Ottawa, Canada, 2019.

C21. Antoine J.-P. Tixier, Maria-Evgenia G. Rossi, Fragkiskos D. Malliaros, Jesse Read, and Michalis Vazirgiannis. Perturb and Combine to Identify Influential Spreaders in Real-World Networks. In Proceedings of the *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (**ASONAM**)*, Vancouver, Canada, 2019.

C20. Adrien Benamira, Benjamin Devillers, Etienne Lesot, Ayush K. Rai, Manal Saadi, and Fragkiskos D. Malliaros. In Proceedings of the *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (**ASONAM**)*, Vancouver, Canada, 2019.

C19. Abdulkadir Çelikkanat and Fragkiskos D. Malliaros. TNE: A Latent Model for Representation Learning on Networks. In Proceedings of the *NeurIPS Relational Representation Learning Workshop (**NeurIPS-R2L**)*, Montréal, Canada, 2018.

C18. Duong Nguyen and Fragkiskos D. Malliaros. BiasedWalk: Biased Sampling for Representation Learning on Graphs. In Proceedings of the *IEEE International Conference on Big Data Workshops (**Big Data**)*, Seattle, WA, 2018.

C17. George Panagopoulos, Fragkiskos D. Malliaros, and Michalis Vazirgiannis. Diffu-Greedy: An Influence Maximization Algorithm based on Diffusion Cascades. In Proceedings of the *International Conference on Complex Networks and Their Applications (**Complex Networks**), Cambridge, UK, 2018*

C16. Konstantinos Skianis, Fragkiskos D. Malliaros, and Michalis Vazirgiannis. Fusing Document, Collection and Label Graph-based Representations with Word Embeddings for Text Classification. In Proceedings of the *12th NAACL-HLT Workshop on Graph-Based Natural Language Processing (**TextGraphs**)*, New Orleans, LA, 2018.
Best paper award.

C15. Ke Sun, Fragkiskos D. Malliaros, Frank Nielsen, and Michalis Vazirgiannis. Reconstructing Uncertain Graphs Based on Low-Rank Factorizations. In Proceedings of the *Conference From Physics to Information Sciences and Geometry (**Entropy**)*, Barcelona, 2018.

C14. Maria-Evgenia G. Rossi, Bowen Shi, Nikolaos Tziortziotis, Fragkiskos D. Malliaros, Christos Giatsidis, and Michalis Vazirgiannis. MATI: An Efficient Algorithm for Influence Maximization in Social Networks. In Proceedings of the *International Conference on Complex Networks and Their Applications (**Complex Networks**), Lyon, France, 2017.*

C13. Kumaran Gunasekaran, Jeyavaishnavi Muralikumar, Sudarshan Srinivasa Ramanujam, Balasubramaniam Srinivasan, and Fragkiskos D. Malliaros. NetGloVe: Learning Node Representations for Community Detection. In Proceedings of the *International Conference on Complex Networks and Their Applications (**Complex Networks**), Lyon, France, 2017.*

C12. Damien Seux, Fragkiskos D. Malliaros, Apostolos Papadopoulos, and Michalis Vazirgiannis. Core Decomposition of Uncertain Graphs Using Representative Instances. In Proceedings of the *International Conference on Complex Networks and Their Applications (**Complex Networks**), Lyon, France, 2017.*

C11. Marc Mitri, Fragkiskos D. Malliaros, and Michalis Vazirgiannis. Sensitivity of Community Structure to Network Uncertainty. In Proceedings of the 2017 *SIAM International Conference on Data Mining (**SDM**)*, Houston, Texas, 2017.

C10. Konstantinos Skianis, Maria-Evgenia G. Rossi, Fragkiskos D. Malliaros, and Michalis Vazirgiannis. SpreadViz: Analytics and Visualization of Spreading Processes in Social Networks. In Proceedings of the *IEEE International Conference on Data Mining (**ICDM**)*, Barcelona, Spain, 2016. (Demo Paper).

C9. Antoine J.-P. Tixier, Fragkiskos D. Malliaros, and Michalis Vazirgiannis. A Graph Degeneracy-based Approach to Keyword Extraction. In Proceedings of the *2016 Conference on Empirical Methods in Natural Language Processing (**EMNLP**)*, Austin, Texas, 2016. (Long Paper).

C8. Fragkiskos D. Malliaros, Konstantinos Skianis, and Michalis Vazirgiannis. Graph-Based Term Weighting for Text Categorization. In Proceedings of the *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (**ASONAM**), Social Media and Risk Workshop*, Paris, France, 2015.

C7. Fragkiskos D. Malliaros and Michalis Vazirgiannis. Disengagement Social Contagion: Assessing Network Vulnerability under Node Departures. In Proceedings of the *International Conference on Computational Social Science* (**ICCSS**), Helsinki, Finland, 2015.

C6. Maria-Evgenia G. Rossi, Fragkiskos D. Malliaros, and Michalis Vazirgiannis. Spread it Good, Spread it Fast: Identification of Influential Nodes in Social Networks. In Proceedings of the *24th International World Wide Web Conference (**WWW**)*, Florence, Italy, 2015.

C5. Christos Giatsidis, Fragkiskos D. Malliaros, Dimitrios M. Thilikos, and Michalis Vazirgiannis. CoreCluster: A Degeneracy Based Graph Clustering Framework. In Proceedings of the *28th AAAI Conference on Artificial Intelligence (**AAAI**)*, Québec City, Canada, 2014.

C4. Fragkiskos D. Malliaros and Michalis Vazirgiannis. To Stay or Not to Stay: Modeling Engagement Dynamics in Social Graphs. In Proceedings of the *22nd ACM International Conference on Information and Knowledge Management (**CIKM**)*, San Francisco, California, 2013. (Full Paper).

C3. Amalia Charisi, Fragkiskos D. Malliaros, Evangelia I. Zacharaki, and Vasileios Megalooikonomou. Multiresolution Similarity Search in Time Series Data: An Application to EEG Signals. In Proceedings of the *6th ACM International Conference on PErvasive Technologies Related to Assistive Environments (**PETRA**)*, Island of Rhodes, Greece, 2013.

C2. Fragkiskos D. Malliaros, Vasileios Megalooikonomou, and Christos Faloutsos. Fast Robustness Estimation in Large Social Graphs: Communities and Anomaly Detection. In Proceedings of the *2012 SIAM International Conference on Data Mining (**SDM**)*, Anaheim, CA, 2012.

C1. Fragkiskos D. Malliaros and Vasileios Megalooikonomou. Expansion Properties of Large Social Graphs. In Proceedings of the *2nd DASFAA International Workshop on Social Networks and Social Media Mining on the Web (**SNSMW**)*, Hong Kong, 2011.

Tutorials

T12. George Panagopoulos[🏛] and Fragkiskos D. Malliaros. Influence Learning and Maximization. In Proceedings of the *13th IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (**ASONAM**)*, Online, 2021.

T11. George Panagopoulos🏛 and Fragkiskos D. Malliaros. Influence Learning and Maximization. In Proceedings of the *21st International Conference on Web Engineering (**ICWE**)*, Online, 2021.

T10. Fragkiskos D. Malliaros, Giannis Nikolentzos, and Michalis Vazirgiannis. GraphRep: Boosting Text Mining, NLP and Information Retrieval with Graphs. In Proceedings of the *ACM International Conference on Information and Knowledge Management (**CIKM**)*, Turin, Italy, 2018.

T9. Fragkiskos D. Malliaros, Polykarpos Meladianos, and Michalis Vazirgiannis. Graph-based Text Representations: Boosting Text Mining, NLP and Information Retrieval with Graphs. In Proceedings of the *The Web Conference (**WWW**)*, Lyon, France, 2018.

T8. Fragkiskos D. Malliaros, Apostolos Papadopoulos, and Michalis Vazirgiannis. Core Decomposition of Networks: Concepts, Algorithms and Applications. In Proceedings of the *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (**ECML-PKDD**)*, Skopje, 2017.

T7. Fragkiskos D. Malliaros and Michalis Vazirgiannis. Graph-based Text Representations: Boosting Text Mining, NLP and Information Retrieval with Graphs. In Proceedings of the *2017 Conference on Empirical Methods in Natural Language Processing (**EMNLP**)*, Copenhagen, Denmark, 2017.

T6. Fragkiskos D. Malliaros, Apostolos Papadopoulos, and Michalis Vazirgiannis. Core Decomposition of Networks: Concepts, Algorithms and Applications. In Proceedings of the *IEEE International Conference on Data Mining (**ICDM**)*, Barcelona, Spain, 2016.

T5. Fragkiskos D. Malliaros, Apostolos Papadopoulos, and Michalis Vazirgiannis. Core Decomposition in Graphs: Concepts, Algorithms and Applications. In Proceedings of the *18th International Conference on Extending Database Technology (**EDBT**)*, Bordeaux, France, 2016.

T4. Fragkiskos D. Malliaros, Michalis Vazirgiannis, and Apostolos Papadopoulos. Core Decomposition: Algorithms and Applications. In Proceedings of the *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (**ASONAM**)*, Paris, France, 2015.

T3. Christos Giatsidis, Fragkiskos D. Malliaros, and Michalis Vazirgiannis. Community Detection and Evaluation in Social and Information Networks (Tutorial). In Proceedings of the *15th International Conference on Web Information System Engineering (**WISE**)*, Thessaloniki, Greece, 2014.

T2. Christos Giatsidis, Fragkiskos D. Malliaros, and Michalis Vazirgiannis. Graph Mining Tools for Community Detection and Evaluation in Social Networks and the Web (Tutorial). In Proceedings of the *22nd International World Wide Web Conference (**WWW**)*, Rio de Janeiro, Brazil, 2013.

T1. Christos Giatsidis, Fragkiskos D. Malliaros, and Michalis Vazirgiannis. Advanced Graph Mining for Community Evaluation in Social Networks and the Web (Tutorial). In Proceedings of the *6th ACM International Conference on Web Search and Data Mining (**WSDM**)*, Rome, Italy, 2013.

B

CURRICULUM VITÆ

LINKS

Website: http://fragkiskos.me/
Google Scholar profile: https://scholar.google.com/citations?user=_7heOKcAAAAJ&hl=en

RESEARCH INTERESTS

Data science, graph-based machine learning, graph representation learning, graph mining , network science, biomedical network analysis

POSITIONS

CentraleSupélec, Paris-Saclay University, France *Oct 2017 - present*

Assistant Professor (Maître de conférences), Centre de Vision Numérique (CVN)
Associate researcher at Inria Saclay, OPIS team
Co-director of the Master in Data Sciences and Business Analytics (CentraleSupélec and ESSEC)
Co-responsible of the 3rd year mention on "Sciences des données et de l'information – Paris-Saclay"

Sapienza University of Rome, Italy *April 2022 - May 2022*

Visiting Professor, Department of Computer, Control & Management Engineering
Mobility grant awarded by the Sapienza University of Rome

University of California San Diego, USA *Oct 2016 - Sep 2017*

Data Science Postdoctoral Scholar
Department of Computer Science and Engineering, Artificial Intelligence group
Member of the Qualcomm Institute and the Information Theory and Applications Center (ITA)

École Polytechnique, France *Oct 2015 - Sep 2016*

Postdoctoral Researcher; Computer Science Laboratory (LIX)

Palo Alto Research Center (PARC), USA *Jun 2014 - Aug 2014*

Research Intern; Interaction and Analytics Laboratory

University of Patras, Greece *Sep 2009 - Jan 2012*

Research Assistant; Multidimensional Data Analysis and Knowledge Management Group

EDUCATION

École Polytechnique, France *Feb 2012 - Sep 2015*

Ph.D. in Computer Science, Laboratoire d'informatique (LIX)
Thesis title: "Mining Social and Information Networks: Dynamics and Applications"
Mention: "Très honorable"
Advisor: Michalis Vazirgiannis, Professor
Jury members: Jean-Loup Guillaume (La Rochelle Université), Jie Tang (Tsinghua University), Christos Faloutsos (CMU), Aristides Gionis (Aalto University), Balázs Kégl (CNRS, Université Paris-Sud), Frank Nielsen (École Polytechnique), Marc Tommasi (Inria, Université Lille 3)

University of Patras, Greece

M.Sc. in Computer Science & Engineering *Oct 2009 - Jul 2011*
Thesis title: "Structural analysis of time-evolving graphs: properties, models and applications"
Advisor: Vasileios Megalooikonomou, Professor
GPA: 9.75/10

Diploma (5-year) in Computer Engineering and Informatics *Oct 2004 - Jul 2009*
GPA: 8.26/10 (ranked 5th out of approx. 270 students)

AWARDS & HONORS

- ◇ Best paper award finalist, AISafery, IJCAI *Aug 2021*
  Workshop in Artificial Intelligence Safety
- ◇ Reviewer award, NeurIPS *2018, 2021*
- ◇ Young researcher grant (JCJC), French National Research Agency (ANR) *Sep 2020*
- ◇ Best paper honorable mention award, ICWSM *Jun 2020*
  International AAAI Conference on Web and Social Media
- ◇ Best paper award, TextGraphs workshop, NAACL *Jun 2018*
  Workshop on Graph-based Algorithms for Natural Language Processing
- ◇ Editor's choice for the complex networks theme, Scientific Reports *May 2018*
  For the article "Locating Influential Nodes in Complex Networks" published in 2016

- ◇ SIAM-NSF Early Career Travel Award *Apr 2017*
  To attend the SIAM International Conference on Data Mining (SDM)
- ◇ Data Science Postdoctoral Fellowship, UC San Diego *Oct 2016*
  Fellowhip from the Qualcomm Institute to conduct independent research (CSE dept.)
- ◇ Thesis Prize (Prix de thèse), École Polytechnique *Jun 2016*
- ◇ Google European Doctoral Fellowship in Graph Mining *Sep 2012*
  15 fellowhips were awarded in Europe, 40 worldwide
- ◇ Best M.Sc. thesis award in Web Science, University of Patras *Apr 2012*

TEACHING EXPERIENCE

Main Instructor (Spring 2017 - present)
DSBA: Master in Data Sciences and Business Analytics, CentraleSupélec and ESSEC Business School
AI: M.Sc. in Artificial Intelligence, CentraleSupélec
CS: Engineering degree, CentraleSupélec

- ◇ **Machine Learning in Network Science**, 3rd year CS, M.Sc. DSBA, AI (24 hours) *2021 - present*
  - ◇ *Spring 2024*, 108 students
  - ◇ *Spring 2023*, 140 students
  - ◇ *Spring 2022*, 133 students
  - ◇ *Spring 2021*, 128 students
- ◇ **Foundations of Machine Learning**, M.Sc. DSBA (24 hours) *2017 - present*
  - ◇ *Fall 2023*, 98 students
  - ◇ *Fall 2022*, 95 students
  - ◇ *Fall 2021*, 91 students
  - ◇ *Fall 2020*, 80 students
  - ◇ *Fall 2019*, 64 students
  - ◇ *Fall 2018*, 76 students
  - ◇ *Fall 2017*, 57 students
- ◇ **Machine Learning**, 2nd year CS (33 hours) *2018 - present*
  - ◇ *Winter 2020-2021*, 224 students
  - ◇ *Winter 2019-2020*, 214 students
  - ◇ *Fall 2019*, 151 students
- ◇ **Mathematical Modeling of Propagation Phenomena**, 1st year CS (21 hours) *2018 - present*
  - ◇ *Winter 2020-2021*, 114 students
  - ◇ *Winter 2019-2020*, 109 students
  - ◇ *Winter 2018-2019*, 102 students
- ◇ **Network Science Analytics**, M.Sc. DSBA, M.Sc. AI, 3rd year CS (24 hours) *2018 - present*
  - ◇ *Spring 2020*, 63 students
  - ◇ *Winter 2019*, 109 students
  - ◇ *Spring 2018*, 57 students
- ◇ **Graph Mining and Network Analysis**, UC San Diego (27 hours) *Spring 2017*
  - ◇ *Graduate-level course (M.S./Ph.D.)*, 57 students

MAIN ADMINISTRATIVE ACTIVITIES

- ◇ Co-director of the Master in Data Sciences and Business Analytics *2019 - present*
  CentraleSupélec and ESSEC Business School. Ranked 3rd worldwide by QS ranking. Responsibilities include student selection and mentoring, curriculum design, scheduling, connection to enterprises. Two-year program: approximately 150 students per year (M1 and M2)
- ◇ Responsible of the mention on "Data and Information Science – Paris-Saclay" *2019 - present*
  Final year (Master 2) specialization at CentraleSupélec's engineering (grande école) curriculum. Responsibilities include student selection and mentoring, curriculum design, scheduling, connection to enterprises. Approximately 60 students pe year.
- ◇ CentraleSupélec Summer School in Artificial Intelligence *2019 - 2023*
  Academic coordinator. Approximately 15 students per year.
- ◇ Faculty recruiting committee, CentraleSupélec *2020 - present*

STUDENTS MENTORED

Ph.D. Students

- Vahan Martirosyan, CentraleSupélec, Paris-Saclay University, Nov '23 - present
  Co-supervised with J. Giraldo and H. Talbot
  Topic: Deep Graph Neural Networks and applications in biomedicine
  Recepient of 3-year Ph.D. Fellowship from the Graduate School (STIC) of Paris-Saclay University
- Yassine Abbahaddou, École Polytechnique, IP Paris, July '22 - present
  Co-supervised with J. Lutzeyer and M. Vazirgiannis
  Topic: Geometric Deep Learning and its application on the insurance sector
- Rajaa El Hamdani, Télécom Paris, IP Paris, Nov '21 - present
  Co-supervised with T. Bonald
  Topic: Robust graph representation learning and applications in misinformation detection
- Alexandre Duval, CentraleSupélec, Paris-Saclay University, Mar '21 - present
  Co-supervised with H. Talbot
  Topic: A graph machine learning journey: from explainability to climate action
  Globalink Graduate Fellowship - Mitacs for a research internship at Mila - Quebec (Jan '22)
- Surabhi Jagtap, CentraleSupélec, Paris-Saclay University, Jan '19 - Feb '23
  Co-supervised with J.-C. Pesquet and L. Duval
  Topic: Multilayer Graph Embeddings for Omics Data Integration in Bioinformatics
  Industrial fellowship from IFP Energies nouvelles
  Currently: Research scientist at CSEM, Switzerland
- George Panagopoulos, École Polytechnique, IP Paris, Jan '19 - Feb '22
  Co-supervised with M. Vazirgiannis
  Topic: Learning Influence Representations : Methods and Applciations
  Currently: Research scientist at Amazon, Luxembourg
- Abdulkadir Çelikkanat, CentraleSupélec, Paris-Saclay University, Dec '17 - April '21
  Co-supervised with N. Paragios
  Title: Graph representation learning with random walk diffusions
  Currently: Assistant Professor at Aalborg University, Denmark

Visiting Ph.D. Students

- Jhony Giraldo, La Rochelle Université, DATAIA PhD mobility grant (Jan '22 - May '22)
  Topic: Understanding over-smoothing and over-squashinng in graph neural networks
  Currently: Assistant Professor at Télécom Paris, IP Paris

M.Sc. Students (master's thesis, research internship, independent study)

- Vahan Martorosyan, Université Paris-Saclay, with J. Giraldo (internship and master's thesis, May '23-Sep '23)
- Nicolas Dunou, Université Paris Dauphine–PSL, with J. Giraldo (internship and master's thesis, May '23-Sep '23)
- Antoine Siraudin, Sorbonne Université, with J.-C. Pesquet (internship and master's thesis, Jun '23-Nov '23)
- Alexandre Duval, CentraleSupélec, (internship and master's thesis), Jun '20-Nov '20
- Chayma Bouzelfa, École Polytechnique de Tunisie, with J.-C. Pesquet (internship and master's thesis, Jun '20-Nov '20)
- George Panagopoulos, École Polytechnique, with M. Vazirgiannis (internship, Jun '18-Dec '18)
- Duong Nguyen, UC San Diego (independent study, Apr '17-Dec '18)
- Balasubramaniam Srinivasan, UC San Diego (independent study, Apr '17-Sep '17)
- Konstantinos Skianis, ENS Cachan (internship and master's thesis, Apr '15-Sep '15)
- Bowen Shi, ENSTA ParisTech and UPMC (internship and master's thesis, Apr '16-Sep '16)
- Laurent Nieuviarts, École Polytechnique (internship and master's thesis, Apr '16-Sep '16)
- Marc Mitri, École Polytechnique (internship and master's thesis, Apr '15-Sep '15)
- Maria-Evgenia Rossi, École Polytechnique (main advisor: M. Vazirgiannis; Feb '14-Nov '17)

Other Students (Semester project, undergrad internship, research track)

- ◇ Théo Saulus (CS, Oct '23-Mar '24)
- ◇ Basile Terver (l'X, Oct '23-Mar '24)
- ◇ Mouad Aderdor (CS, Oct '22-Mar '23)
- ◇ Simon Rouard (CS, Oct '21-Mar '22)
- ◇ Elias Aouad (CS, Oct '20-Mar '21)
- ◇ Yassine Abbahaddou (CS, Oct '20-Mar '21)
- ◇ Othmane Aitboumlik (CS, Oct '20-Mar '21),
- ◇ Mohamed Ali Kammoun (ENSTA, Jun '19-Aug '19)
- ◇ Tarmach Oumani (CS, Oct '19-Mar '20)
- ◇ Simon Brandeis (CS, Oct '19-Mar '20
- ◇ Pierre Sevestre (CS, Oct '19-Mar '20)
- ◇ Benjamin Deviller (CS, Oct '18-Mar '19)
- ◇ Mehdi Boubnan (CS, Oct '18-Mar '19)
- ◇ Tommaso Chiara (CS, Oct '18-Mar '19)
- ◇ Lorenzo Gasparollo (CS, Oct '18-Mar '19)
- ◇ Xiaoyu Sun (CS, Dec '17-May '19)
- ◇ Ali Ramlaoui (CS, Oct '23-Mar '24)
- ◇ Salah Eddine Azekour (CS, Oct '22-Mar '23)
- ◇ Antonin Gagneré, CS (Oct '21-Mar '22)
- ◇ Hugo Schnoering, CS (Oct '20-Mar '21)
- ◇ Othmane Jebbari (CS, Oct '20-Mar '21)
- ◇ Rémi Castera (l'X, May '20-Aug '20)
- ◇ Randa Elmrabet (CS, Oct '19-Mar '20)
- ◇ Yassine Yahyaoui (CS, Oct '19-Mar '20)
- ◇ Adrian Jarret (CS, Oct '19-Mar '20)
- ◇ Adrien Benamira(CS, Oct '18-Mar '19)
- ◇ Houssam Zenati (CS, Oct '18-Mar '19)
- ◇ Otmane Sakhi (CS, Oct '18-Mar '19)
- ◇ Adrien Seguret (CS, Oct '18-Mar '19)
- ◇ Etienne Lesot (CS, Oct '18-Mar '19)
- ◇ Damien Seux (l'X, Nov '15-Mar '16)

Research Engineers

- ◇ Marc Mitri (Jan '20 - Apr '20)

Dissertation Committee Member

- ◇ Andrea Mastropietro, Sapienza Univerity of Rome (advised by: Aris Anagnostopoulos, expected defense date: Feb '24)
- ◇ Arpit Merchant, University of Helsinki (advised by: Michael Mathioudakis, defense date: Sep '20)
- ◇ Jun Zhu, CentraleSupélec, Paris-Saclay University (advised by: Céline Hudelot and Paul-Henry Cournède, defense date: Mar '23)
- ◇ Jhony Giraldo, La Rochelle Université (advised by: Thierry Bouwmans, defense date: Sept '22)
- ◇ Rongyan Zhou, CentraleSupélec. Paris-Saclay University (advised by: Julie Le Cardinal, defense date: Apr '21)
- ◇ Elham Alghamdi, University College Dublin (advised by: Derek Greene, defense date: Nov '20)
- ◇ Choe Adam, CentraleSupélec, Paris-Saclay University (advised by: Paul-Henry Cournède, defense date: Apr '19)

FUNDING

- ◇ **ANR RHU grant** (Recherche Hospitalo-Universitaire en Sante) Mar 2024 - Feb 2029
  INNOV4-ePiK: Innovative diagnostic and therapeutic approaches in potassium channel developmental and epileptic encephalopathies (K-DEEs) using 4P for medicine
  PI: Rima Nabbout (Imagine Institute of Genetic Diseases); co-PI: Fragkiskos Malliaros (11 partners in total)
  Total: 340,000€
- ◇ **Visiting professor mobility grant, Sapienza University of Rome** Apr 2022 - May 2022
  Network Science Analytics for the Internet of Things
  PI: Ioannis Chatzigiannakis (Sapienza Univ. of Rome); co-PI: Fragkiskos Malliaros
  Total: 5,000€
- ◇ **DATAIA Institute** Jan 2022 - May 2022
  Mobility grant to support the visit of a Ph.D. student to CentraleSupélec
  PI: Fragkiskos Malliaros
  Total 5,000€
- ◇ **Labex DigiCosme** Nov 2021 - Oct 2024
  GratifAI: Graph Enhancement for Robust Representation Learning and Applications
  Type: Ph.D. grant
  PI: Fragkiskos Malliaros, Thomas Bonald (Télécom Paris)
  Total: 120,000€
- ◇ **ANR JCJC grant** (French National Research Agency) Mar 2021 - Feb 2025
  GraphIA: Scalable and Robust Representation Learning on Graphs

PI: Fragkiskos Malliaros
Total: 225,000€

◇ **Thales Group** Oct 2020 - Sep 2023
Neural Networks for Safety of Complex Systems
Type: Ph.D. grant
PI: Jean-Christophe Pesquet (CentraleSupélec); co-PI: Fragkiskos Malliaros
Total: 150,000€

◇ **DATAIA Institute** 2019 - 2022
UltraBioLearn: Machine Learning for the Identification and Validation of Prognostic and Predictive Biomarkers in Immunotherapy
PI: Hugues Talbot (CentraleSupélec); co-PI: Nathalie Lassau (Université Paris-Saclay), Fragkiskos Malliaros
Total: 212,000€

◇ **SNCF** (French National Railway Company) Nov 2019 - Mar 2020
PLATIPUS: Preliminary Study of Machine Learning to Determine the Scouring Risk of SNCF's Infrastructure
PI: Fragkiskos Malliaros, Maria Vakalopoulou (CentraleSupélec)
Total: 20,000€

◇ **CentraleSupélec Research Department** Jun 2019 - Jul 2019
Grant to support the scientific visit of invited professor
PI: Fragkiskos Malliaros; co-PI: Apostolos Papadopoulos (Aristotle Univ. of Thessaloniki)
Total: 6,400€

◇ **IFP Energies nouvelles** Jan 2019 - Dec 2021
Graph-based Learning from Integrated Multi-omics and Multi-species Data
Type: Ph.D. grant
PI: Fragkiskos Malliaros; co-PI: Jean-Christophe Pesquet (CentraleSupélec)
Total: 150,000€

◇ **SNCF – Railenium** 2018 - 2019
SIARA: A Fully Automatic, Accurate and Efficient System for Monitoring the Railway Network
PI: Fragkiskos Malliaros, Maria Vakalopoulou (CentraleSupélec), Nikos Paragios (CentraleSupélec)
Total: 180,000€

Selected Invited Talks

◇ *Graph Machine Learning and Biomedical Application*
○ Board of European Students of Technology (BEST), France *April 2023*

◇ *Graph Representation Learning: Matrix Factorization and Random Walks*
○ DTU Compute, Technical University of Denmark *Aug 2022*

◇ *Learning Graph Representations with Random Walks: Models and Applications*
○ Dept. of Computer, Control and Management Eng., Sapienza University of Rome, Italy *May 2022*
○ Aristotle University of Thessaloniki, Greece (Online) *Apr 2021*

◇ *Machine Learning on Heterogeneous, Text-Rich Information Networks*
○ Rakuten Institute of Technology, Rakuten Inc., France (Online) *Nov 2020*

◇ *Machine Learning in Network Science and Applications*
○ Booking.com, The Netherlands (Online) *May 2020*
○ IFP Energies nouvelles, France *Mar 2020*
○ Congrès Scientifique du Campus de Saclay, France *Mar 2019*

◇ *Mining Social and Information Networks*
○ Paris Descartes University, Paris, France *May 2018*
○ AI Seminar, UC San Diego, San Diego, CA *Oct 2016*
○ Télécom ParisTech, Paris, France *Jun 2016*
○ Yale School of Management, Yale University, New Haven, CT (Online) *Apr 2016*
○ NEC Laboratories Europe, Heidelberg, Germany *Apr 2016*
○ Senseable City Laboratory, MIT, Cambridge, MA (Online) *Mar 2016*

◇ *Degeneracy-Based Mining of Social and Information Networks*
○ Complex Networks Team, Pierre and Marie Curie University, Paris, France *Apr 2016*
○ CCNR, Network Science Institute, Northeastern University, Boston, MA *Mar 2016*

◇ *Mining Social and Information Networks: Dynamics and Applications*
- ○ Computational Social Science, ETH Zürich (Online) *Dec 2015*
- ○ Bell Labs, Alcatel-Lucent, Dublin, Ireland *Oct 2015*

◇ *Predicting Quitting Behavior in Enterprise Social Networks*
- ○ Palo Alto Research Center (PARC), Palo Alto, CA *Aug 2014*

◇ *To Stay or Not to Stay: Modeling Engagement Dynamics in Social Graphs*
- ○ Google Zürich, Doctoral Fellowship Forum *Sep 2013*

Professional Service

Journal Editor

◇ Associate Editor, Elsevier Big Data Research *2024 - present*

◇ Guest Editor, International Journal of Data Science and Analytics (JDSA), Springer *2024*
Special issue on *Theoretical and Practical Data Science and Analytics*

Organizing Committee

◇ Journal track co-chair, IEEE DSAA 2024, San Diego, CA, 2024
11th IEEE International Conference on Data Science and Advanced Analytics

◇ Special session on "GraDSci: Graph Data Science and Applications"
10th IEEE International Conference on Data Science and Advanced Analytics (DSAA), Thessaloniki, Greece, 2023

◇ Demonstration co-chair, ECML-PKDD, Grenoble, France, 2022

◇ 15th International Workshop on Graph-Based Natural Language Processing (TextGraphs)
Co-located with *NAACL*, Mexico City, 2021

◇ 14th International Workshop on Graph-Based Natural Language Processing (TextGraphs)
Co-located with *COLING*, Barcelona, 2020

◇ Special session on "Machine Learning with Graphs: Algorithms and Applications"
International Conference on Artificial Neural Networks 2019 (ICANN), Munich, 2019

◇ 4th International Workshop on Deep Learning for Graphs and Structured Data Embedding
Co-located with *The Web Conference (WWW)*, San Francisco, CA, 2019

◇ 3rd International Workshop on Learning Representations for Big Networks (BigNet)
Co-located with *The Web Conference (WWW)*, Lyon, France, 2018

◇ 2017 Information Theory and Applications Workshop (ITA), San Diego, CA, 2017

◇ Special Session on "Natural Language Processing for Social Media Analysis"
Co-located with the *19th International Conference on Speech and Computer (SPECOM)*, Hatfield, UK, 2017

◇ 2nd International Workshop on Data Science for Social Media and Risk (SoMeRiS)
Co-located with the *IEEE International Conference on Data Mining (ICDM)*, Barcelona, Spain, 2016

Recent Program Committee Member

International Conference on Learning Representations (**ICLR**), 2021-present; International Conference on Machine Learning (**ICML**), 2019 - present; Conference on Neural Information Processing Systems (**NeurIPS**), 2018 - present; The Web Conference (**WWW**): 2018 - present; AAAI Conference on Artificial Intelligence (**AAAI**): 2017, 2018, 2020, 2021; Internation Conference on Web and Social Media (**ICWSM**): 2019, 2021; European Conference on Machine Learning (**ECML-PKDD**), 2020 - present

Journal Reviewer

ACM Transactions on Knowledge Discovery from Data (TKDD), IEEE Transactions on Knowledge and Data Engineering (TKDE), Data Mining and Knowledge Discovery (DAMI), Applied Network Science, Springer, Computer Vision and Image Understanding (CVIU), Social Network Analysis and Mining (SNAM), Intelligent Data Analysis (IDA), Information Retrieval (Springer), Pattern Recognition (Elsevier)

BIBLIOGRAPHY

[Ahm+13] Amr Ahmed, Nino Shervashidze, Shravan Narayanamurthy, Vanja Josifovski, and Alexander J. Smola. Distributed Large-Scale Natural Graph Factorization. In *WWW*, Association for Computing Machinery, 2013, 37–48 (cited on page 8).

[AY21a] Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. In *ICLR*, 2021 (cited on pages 48, 50, 53).

[AY21b] Qi An and Liang Yu. A heterogeneous network embedding framework for predicting similarity-based drug-target interactions. *Briefiengs in Bioinformatics* 22:6 (2021), pp. 1–10 (cited on pages 40, 44).

[And70] Erling Andersen. Sufficiency and Exponential Families for Discrete Sample Spaces. *Journal of the American Statistical Association*. 65:331 (1970), pp. 1248–1255 (cited on page 13).

[AD18] Sinan Aral and Paramveer S Dhillon. Social influence maximization under empirical influence models. *Nature Human Behaviour* 2:6 (2018), p. 375 (cited on page 73).

[Are+08] Alex Arenas, Alberto Fernández, Santo Fortunato, and Sergio Gómez. Motif-based communities in complex networks. *J. Phys. A Math. Theor.* 41:22 (2008), p. 224001 (cited on pages 55, 56).

[BKH21] Jinheon Baek, Minki Kang, and Sung Ju Hwang. Accurate Learning of Graph Representations with Graph Multiset Pooling. In *ICLR*, 2021 (cited on page 61).

[BK18] Arunkumar Bagavathi and Siddharth Krishnan. Multi-net: a scalable multiplex network embedding framework. In *Complex Networks*, 2018, pp. 119–131 (cited on page 37).

[Bag+21] Maryam Bagherian, Elyas Sabeti, Kai Wang, Maureen A. Sartor, Zaneta Nikolovska-Coleska, and Kayvan Najarian. Machine learning approaches and databases for prediction of drug-target interaction: A survey paper. *Briefiengs in Bioinformatics* 22:1 (2021), pp. 247–269 (cited on page 40).

[Bak+11] Eytan Bakshy, Jake M Hofman, Winter A Mason, and Duncan J Watts. Everyone's an influencer: quantifying influence on twitter. In *WSDM*, 2011 (cited on page 78).

[BA99] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science* 286:5439 (1999), pp. 509–512 (cited on page 83).

[BBM12] Nicola Barbieri, Francesco Bonchi, and Giuseppe Manco. Topic-Aware Social Influence Propagation Models. In *ICDM*, 2012 (cited on page 85).

[Bat+20] Federico Battiston, Giulia Cencetti, Iacopo Iacopini, Vito Latora, Maxime Lucas, Alice Patania, Jean-Gabriel Young, and Giovanni Petri. Networks beyond pairwise interactions: Structure and dynamics. *Physics Reports* 874 (2020). Networks beyond pairwise interactions: Structure and dynamics, pp. 1–92 (cited on page 56).

[BN01] Mikhail Belkin and Partha Niyogi. Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering. In *NIPS*, Cambridge, MA, USA, 2001, pp. 585–591 (cited on page 8).

[BGL16] Austin R Benson, David F Gleich, and Jure Leskovec. Higher-order organization of complex networks. *Science* 353:6295 (2016), pp. 163–166 (cited on pages 56, 59).

[Ber+13] Michele Berlingerio, Michele Coscia, Fosca Giannotti, Anna Monreale, and Dino Pedreschi. Multidimensional networks: foundations of structural analysis. *World Wide Web* 16:5-6 (2013), pp. 567–593 (cited on page 31).

[Bho+20] Ayan Kumar Bhowmick, Koushik Meneni, Maximilien Danisch, Jean-Loup Guillaume, and Bivas Mitra. LouvainNE: Hierarchical Louvain Method for High Quality and Scalable Network Embedding. In *WSDM*, 2020, pp. 43–51 (cited on pages 24, 25, 27).

[BGA20] Filippo Maria Bianchi, Daniele Grattarola, and Cesare Alippi. Spectral clustering with graph neural networks for graph pooling. In *ICML*, 2020 (cited on pages 59, 61).

[BNJ03] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research.* 3:1 (2003), pp. 993–1022 (cited on page 10).

[Blo+08] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *J. Stat. Mech.* 2008:10 (2008), P10008 (cited on pages 10, 20).

[Boc+14] S. Boccaletti, G. Bianconi, R. Criado, C.I. del Genio, J. Gómez-Gardeñes, M. Romance, I. Sendiña-Nadal, Z. Wang, and M. Zanin. The structure and dynamics of multilayer networks. *Physics Reports* 544:1 (2014), pp. 1–122 (cited on pages 3, 31, 37).

[BLG16] Simon Bourigault, Sylvain Lamprier, and Patrick Gallinari. Representation learning for information diffusion through social networks: an embedded cascade model. In *WSDM*, 2016 (cited on pages 72, 73, 75).

[BM+22] Luc Brogat-Motte, Rémi Flamary, Celine Brouard, Juho Rousu, and Florence D'Alché-Buc. Learning to Predict Graphs with Fused Gromov-Wasserstein Barycenters. In *ICML*, 2022, pp. 2321–2335 (cited on page 88).

[Bro+21] Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges. *CoRR* (2021) (cited on pages 4, 47).

[BBS11] Céline Brouard, Florence d'Alché Buc, and Marie Szafranski. Semi-Supervised Penalized Output Kernel Regression for Link Prediction. In *ICML*, 2011 (cited on page 17).

[BAEA11] Ceren Budak, Divyakant Agrawal, and Amr El Abbadi. Limiting the Spread of Misinformation in Social Networks. In *WWW*, 2011 (cited on page 71).

[CLX15] Shaosheng Cao, Wei Lu, and Qiongkai Xu. GraRep: Learning Graph Representations with Global Structural Information. In *CIKM*, 2015, pp. 891–900 (cited on pages 8, 16, 17, 24).

[Car97] Rich Caruana. Multitask learning. *Machine Learning* 28:1 (1997), pp. 41–75 (cited on page 74).

[CC+23] Jhon A. Castro-Correa, Jhony H. Giraldo, Anindya Mondal, Mohsen Badiey, Thierry Bouwmans, and Fragkiskos D. Malliaros. Time-Varying Signals Recovery Via Graph Neural Networks. In *ICASSP*, 2023 (cited on page 88).

[ÇM20] Abdulkadir Çelikkanat and Fragkiskos D. Malliaros. Exponential Family Graph Embeddings. In *AAAI*, 2020, pp. 3357–3364 (cited on pages 3, 7, 12, 20).

[ÇM19] Abdulkadir Çelikkanat and Fragkiskos D. Malliaros. Kernel Node Embeddings. In *GlobalSIP*, 2019, pp. 1–5 (cited on pages 3, 7, 16).

[ÇM21] Abdulkadir Çelikkanat and Fragkiskos D. Malliaros. Topic-Aware Latent Models for Representation Learning on Networks. *Pattern Recognition Letters* 144 (2021), pp. 89–96 (cited on pages 2, 3, 7, 9, 20).

[ÇMP22] Abdulkadir Çelikkanat, Fragkiskos D. Malliaros, and Apostolos N. Papadopoulos. NodeSig: Binary Node Embeddings via Random Walk Diffusion. In *ASONAM*, 2022, pp. 68–75 (cited on pages 3, 7, 23).

[ÇSM23] Abdulkadir Çelikkanat, Yanning Shen, and Fragkiskos D. Malliaros. Multiple Kernel Representation Learning on Networks. *IEEE Trans. Knowl. Data Eng.* 35:6 (2023), pp. 6113–6125 (cited on pages 3, 7, 16, 20).

[CF06] Deepayan Chakrabarti and Christos Faloutsos. Graph Mining: Laws, Generators, and Algorithms. *ACM Comput. Surv.* 38:1 (2006), 2–es (cited on page 89).

[CL11] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A Library for Support Vector Machines. *ACM Trans. Intell. Syst. Technol.* 2:3 (2011) (cited on page 38).

[Cha02] Moses S. Charikar. Similarity Estimation Techniques from Rounding Algorithms. In *STOC*, 2002, 380–388 (cited on page 26).

[Che70] Jeff Cheeger. A lower bound for the smallest eigenvalue of the Laplacian. *Problems in Analysis* 625:195-199 (1970), p. 110 (cited on page 49).

[Che+12] Duanbing Chen, Linyuan Lü, Ming-Sheng Shang, Yi-Cheng Zhang, and Tao Zhou. Identifying influential nodes in complex networks. *Physica A: Statistical Mechanics and its Applications* 391:4 (2012), pp. 1777–1787 (cited on page 72).

[Che+18] Haochen Chen, Bryan Perozzi, Yifan Hu, and S. Skiena. HARP: Hierarchical Representation Learning for Networks. In *AAAI*, 2018 (cited on page 24).

[Che+19] Haochen Chen, Syed Fahad Sultan, Yingtao Tian, Muhao Chen, and Steven Skiena. Fast and Accurate Network Embeddings via Very Sparse Random Projection. In *CIKM*, 2019, pp. 399–408 (cited on page 24).

[CWY09] Wei Chen, Yajun Wang, and Siyu Yang. Efficient influence maximization in social networks. In *KDD*, 2009 (cited on page 84).

[CBP16] Hyunghoon Cho, Bonnie Berger, and Jian Peng. Compact integration of multi-network topology for functional analysis of genes. *Cell systems* 3:6 (2016), pp. 540–548 (cited on page 37).

[Chu97] Fan R. K. Chung. *Spectral graph theory*. 92. American Mathematical Soc., 1997 (cited on pages 47, 49, 50).

[Con04] Gene Ontology Consortium. The Gene Ontology (GO) database and informatics resource. *Nucleic acids research* 32:suppl_1 (2004), pp. D258–D261 (cited on page 38).

[DBV16] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *NIPS*, 2016 (cited on pages 48, 55).

[DGK04] Inderjit S. Dhillon, Yuqiang Guan, and Brian Kulis. Kernel K-means: Spectral Clustering and Normalized Cuts. In *KDD*, 2004, pp. 551–556 (cited on page 16).

[DG+23] Francesco Di Giovanni, Lorenzo Giusti, Federico Barbero, Giulia Luise, Pietro Lio, and Michael Bronstein. On Over-Squashing in Message Passing Neural Networks: The Impact of Width, Depth, and Topology. In *ICML*, 2023 (cited on page 50).

[DTG20] Yijie Ding, Jijun Tang, and Fei Guo. Identification of drug–target interactions via dual laplacian regularized least squares with multiple kernel fusion. *Knowledge Based Systems* 204 (2020), p. 106254 (cited on pages 40, 44).

[Don+12] Xiaowen Dong, Pascal Frossard, Pierre Vandergheynst, and Nikolai Nefedov. Clustering with multi-layer graphs: A spectral perspective. *IEEE Transactions on Signal Processing* 60:11 (2012), pp. 5820–5831 (cited on page 37).

[DOT14] Kimon Drakopoulos, Asuman E. Ozdaglar, and John N. Tsitsiklis. An Efficient Curing Policy for Epidemics on Graphs. *IEEE Trans. Netw. Sci. Eng.* 1:2 (2014), pp. 67–75 (cited on page 71).

[Du+14] Nan Du, Yingyu Liang, Maria Balcan, and Le Song. Influence function learning in information diffusion networks. In *ICML*, 2014 (cited on page 73).

[Du+13] Nan Du, Le Song, Manuel Gomez Rodriguez, and Hongyuan Zha. Scalable influence estimation in continuous-time diffusion networks. In *NeurIPS*, 2013 (cited on pages 73, 74, 78).

[DM21] Alexandre Duval and Fragkiskos D. Malliaros. GraphSVX: Shapley Value Explanations for Graph Neural Networks. In *ECML PKDD*, 2021, pp. 302–318 (cited on pages 3, 4, 47, 62, 63, 67).

[DM22] Alexandre Duval and Fragkiskos D. Malliaros. Higher-order Clustering and Pooling for Graph Neural Networks. In *CIKM*, 2022, pp. 426–435 (cited on pages 3, 4, 47, 55, 58, 61).

[Duv+23a] Alexandre Duval, Simon V. Mathis, Chaitanya K. Joshi, Victor Schmidt, Santiago Miret, Fragkiskos D. Malliaros, Taco Cohen, Pietro Lio, Yoshua Bengio, and Michael Bronstein. A Hitchhiker's Guide to Geometric GNNs for 3D Atomic Systems. *CoRR* (2023) (cited on page 89).

[Duv+23b] Alexandre Duval, Victor Schmidt, Alex Hernández-García, Santiago Miret, Fragkiskos D. Malliaros, Yoshua Bengio, and David Rolnick. FAENet: Frame Averaging Equivariant GNN for Materials Modeling. In *ICML*, 2023, pp. 9013–9033 (cited on page 89).

[Dwi+22] Vijay Prakash Dwivedi, Ladislav Rampášek, Mikhail Galkin, Ali Parviz, Guy Wolf, Anh Tuan Luu, and Dominique Beaini. Long range graph benchmark. In *NeurIPS*, 2022 (cited on page 48).

[EP04] Theodoros Evgeniou and Massimiliano Pontil. Regularized multi–task learning. In *KDD*, 2004 (cited on page 75).

[Ezz+17] Ali Ezzat, Peilin Zhao, Min Wu, Xiao Li Li, and Chee Keong Kwoh. Drug-target interaction prediction with graph regularized matrix factorization. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 14:3 (2017), pp. 646–656 (cited on page 41).

[Fan+20] Changjun Fan, Li Zeng, Yizhou Sun, and Yang-Yu Liu. Finding key players in complex networks through deep reinforcement learning. *Nature Machine Intelligence* (2020), pp. 1–8 (cited on page 84).

[FGZ20] Kunjie Fan, Yuanfang Guan, and Yan Zhang. Graph2GO: a multi-modal attributed network embedding method for inferring protein functions. *GigaScience* 9:8 (2020), giaa081 (cited on page 37).

[Far+17] Mehrdad Farajtabar, Jiachen Yang, Xiaojing Ye, Huan Xu, Rakshit Trivedi, Elias Khalil, Shuang Li, Le Song, and Hongyuan Zha. Fake News Mitigation via Point Process Based Intervention. In *ICML*, 2017 (cited on page 71).

[Fen+18] Shanshan Feng, Gao Cong, Arijit Khan, Xiucheng Li, Yong Liu, and Yeow Meng Chee. Inf2vec: Latent Representation Model for Social Influence Embedding. In *ICDE*, 2018 (cited on pages 72, 74, 75, 78).

[Fey+18] Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Heinrich Müller. Splinecnn: Fast geometric deep learning with continuous b-spline kernels. In *CVPR*, 2018 (cited on page 55).

[For10] Santo Fortunato. Community detection in graphs. *Physics Reports* 486:3 (2010), pp. 75 –174 (cited on pages 9, 56, 60).

[GWG19] Johannes Gasteiger, Stefan Weißenberger, and Stephan Günnemann. Diffusion improves graph learning. In *NeurIPS*, 2019 (cited on pages 48, 53).

[Gia+14] Christos Giatsidis, Fragkiskos D. Malliaros, Dimitrios M. Thilikos, and Michalis Vazirgiannis. CoreCluster: A Degeneracy Based Graph Clustering Framework. In *AAAI*, 2014 (cited on page 2).

[Gia+16] Christos Giatsidis, Fragkiskos D. Malliaros, Nikolaos Tziortziotis, Charanpal Dhanjal, Emmanouil Kiagias, Dimitrios M. Thilikos, and Michalis Vazirgiannis. A *k*-core Decomposition Framework for Graph Clustering. *CoRR* (2016) (cited on page 2).

[Gil+17] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural Message Passing for Quantum Chemistry. In *ICML*, 2017 (cited on page 47).

[Gir+23] Jhony H. Giraldo, Konstantinos Skianis, Thierry Bouwmans, and Fragkiskos D. Malliaros. On the Trade-off between Over-Smoothing and Over-Squashing in Deep Graph Neural Networks. In *CIKM*, 2023 (cited on pages 3, 4, 47, 48, 53).

[GN02] M. Girvan and M. E. J. Newman. Community Structure in Social and Biological Networks. *PNAS* 99:12 (2002), pp. 7821–7826 (cited on page 9).

[GBB18] Vladimir Gligorijević, Meet Barot, and Richard Bonneau. deepNF: deep network fusion for protein function prediction. *Bioinformatics* 34:22 (2018), pp. 3873–3881 (cited on page 37).

[GW95] Michel X. Goemans and David P. Williamson. Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming. *J. ACM* 42 (1995) (cited on page 26).

[GA11] Mehmet Gönen and Ethem Alpaydın. Multiple Kernel Learning Algorithms. *J. Mach. Learn. Res.* 12 (2011) (cited on pages 17, 19).

[GBL11] Amit Goyal, Francesco Bonchi, and Laks V. S. Lakshmanan. A Data-Based Approach to Social Influence Maximization. *Proc. VLDB Endow.* 5:1 (2011), pp. 73–84 (cited on pages 72, 78).

[GBL10] Amit Goyal, Francesco Bonchi, and Laks VS Lakshmanan. Learning influence probabilities in social networks. In *WSDM*, 2010 (cited on pages 72, 74, 78).

[GLL11a] Amit Goyal, Wei Lu, and Laks V.S. Lakshmanan. CELF++: Optimizing the Greedy Algorithm for Influence Maximization in Social Networks. In *WWW*, 2011, 47–48 (cited on page 72).

[GLL11b] Amit Goyal, Wei Lu, and Laks VS Lakshmanan. Simpath: An efficient Algorithm for Influence Maximization under the Linear Threshold model. In *ICDM*, 2011 (cited on page 78).

[GL16] Aditya Grover and Jure Leskovec. Node2Vec: Scalable Feature Learning for Networks. In *KDD*, 2016, pp. 855–864 (cited on pages 2, 21, 25).

[Gui+18] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A Survey of Methods for Explaining Black Box Models. *ACM Comput. Surv.* 51:5 (2018) (cited on page 62).

[GZ23] Xiaojie Guo and Liang Zhao. A Systematic Survey on Deep Generative Models for Graph Generation. *IEEE Trans. Pattern Anal. Mach. Intell.* 45:5 (2023), pp. 5370–5390 (cited on page 89).

[Gur+22] Saket Gurukar, Priyesh Vijayan, srinivasan parthasarathy, Balaraman Ravindran, Aakash Srinivasan, Goonmeet Bajaj, Chen Cai, Moniba Keymanesh, Saravana Kumar, Pranav Maneriker, Anasua Mitra, and Vedang Patel. Benchmarking and Analyzing Unsupervised Network Representation Learning and the Illusion of Progress. *Transactions on Machine Learning Research* (2022) (cited on page 29).

[Ham20] William L. Hamilton. *Graph Representation Learning*. Morgan and Claypool Publishers, 2020 (cited on pages 1, 4, 7, 24, 47).

[HYL17a] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive Representation Learning on Large Graphs. In *NIPS*, 2017 (cited on page 48).

[HYL17b] William L. Hamilton, Rex Ying, and Jure Leskovec. Representation Learning on Graphs: Methods and Applications. *IEEE Data Eng. Bull.* 40:3 (2017), pp. 52–74 (cited on pages 7, 8).

[HSS08] Thomas Hofmann, Bernhard Schölkopf, and Alexander J. Smola. Kernel Methods in Machine Learning. *The Annals of Statistics* 36:3 (2008), pp. 1171–1220 (cited on pages 16, 18).

[HK02] P. Holme and B. J. Kim. Growing scale-free networks with tunable clustering. *Phys. Rev. E* 65:2 (2002), p. 026107 (cited on page 83).

[Hua+23] Qiang Huang, Makoto Yamada, Yuan Tian, Dinesh Singh, and Yi Chang. GraphLIME: Local Interpretable Model Explanations for Graph Neural Networks. *EEE Trans. Knowl. Data Eng.* 35:7 (2023), pp. 6968–6972 (cited on pages 62, 66).

[Jag+22a] Surabhi Jagtap, Aurélie Pirayre, Frédérique Bidard, Laurent Duval, and Fragkiskos D. Malliaros. BRANEnet: Embedding Multilayer Networks for Omics Data Integration. *BMC Bioinformatics* 23:1 (2022), p. 429 (cited on pages 3, 4, 31, 35, 37).

[Jag+21a] Surabhi Jagtap, Aurélie Pirayre, Frédérique Bidard, Laurent Duval, and Fragkiskos D. Malliaros. BRANet: Graph-based Integration of Multi-omics Data with Biological a priori for Regulatory Network Inference. In *CIBB*, 2021 (cited on pages 31, 35).

[Jag+21b] Surabhi Jagtap, Abdulkadir Çelikkanat, Aurélie Pirayre, Frédérique Bidard, Laurent Duval, and Fragkiskos D. Malliaros. Multiomics Data Integration for Gene Regulatory Network Inference with Exponential Family Embeddings. In *EUSIPCO*, 2021 (cited on pages 3, 4, 31, 33, 37).

[Jag+22b] Surabhi Jagtap, Abdulkadir Çelikkanat, Aurélie Pirayre, Frédérique Bidard, Laurent Duval, and Fragkiskos D Malliaros. BraneMF: Integration of Biological Networks for Functional Analysis of Proteins. *Bioinformatics* 38:24 (2022), pp. 5383–5389 (cited on pages 3, 4, 31, 35, 37).

[Jin+23] Ming Jin, Huan Yee Koh, Qingsong Wen, Daniele Zambon, Cesare Alippi, Geoffrey I. Webb, Irwin King, and Shirui Pan. A Survey on Graph Neural Networks for Time Series: Forecasting, Classification, Imputation, and Anomaly Detection. *CoRR* (2023) (cited on page 88).

[JLS86] William B. Johnson, Joram Lindenstrauss, and Gideon Schechtman. Extensions of lipschitz maps into Banach spaces. *Israel Journal of Mathematics* 54 (1986), pp. 129–138 (cited on page 25).

[JL14] Jürgen Jost and Shiping Liu. Ollivier's Ricci curvature, local clustering and curvature-dimension inequalities on graphs. *Discrete & Computational Geometry* 51:2 (2014), pp. 300–322 (cited on page 52).

[KBM23] Kedar Karhadkar, Pradeep Kr Banerjee, and Guido Montúfar. FoSR: First-order spectral rewiring for addressing oversquashing in GNNs. In *ICLR*, 2023 (cited on pages 53, 54).

[KKT03] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *KDD*, 2003 (cited on pages 71, 72, 77, 80).

[KW17] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*, 2017 (cited on pages 48, 51, 53, 67).

[Kit+10] M. Kitsak, L. Gallos, S. Havlin, F. Liljeros, L. Muchnik, H. Stanley, and H. Makse. Identification of Influential Spreaders in Complex Networks. *Nature Physics* 6:11 (2010), pp. 888–893 (cited on pages 72, 77, 84).

[Kiv+14] Mikko Kivelä, Alex Arenas, Marc Barthelemy, James P. Gleeson, Yamir Moreno, and Mason A. Porter. Multilayer networks. *Journal of Complex Networks* 2:3 (2014), pp. 203–271 (cited on pages 3, 31).

[KL02] Risi Imre Kondor and John Lafferty. Diffusion kernels on graphs and other discrete input spaces. In *ICML*, 2002, pp. 315–322 (cited on page 17).

[KJM20] Nils M. Kriege, Fredrik D. Johansson, and Christopher Morris. A survey on graph kernels. *Appl. Netw. Sci.* 5:1 (2020) (cited on page 17).

[Lag+18] Paul Lagrée, Olivier Cappé, Bogdan Cautis, and Silviu Maniu. Algorithms for Online Influencer Marketing. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 13:1 (2018), p. 3 (cited on page 73).

[LFR08] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. Benchmark graphs for testing community detection algorithms. *Phys. Rev. E* 78 (4 2008), p. 046110 (cited on page 89).

[Lee+19] John Boaz Lee, Ryan A. Rossi, Xiangnan Kong, Sungchul Kim, Eunyee Koh, and Anup Rao. Graph Convolutional Networks with Motif-based Attention. In *CIKM*, 2019 (cited on page 56).

[LLK19] Junhyun Lee, Inyeop Lee, and Jaewoo Kang. Self-attention graph pooling. In *ICML*, 2019 (cited on page 61).

[LG10] Kristina Lerman and Rumi Ghosh. Information Contagion: An Empirical Study of the Spread of News on Digg and Twitter Social Networks. In *ICWSM*, 2010 (cited on page 77).

[Les+07] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. Cost-effective outbreak detection in networks. In *KDD*, 2007 (cited on pages 72, 77, 79, 81).

[LK14] Jure Leskovec and Andrej Krevl. *SNAP Datasets: Stanford Large Network Dataset Collection*. http://snap.stanford.edu/data. June 2014 (cited on pages 21, 27).

[Les+09] Jure Leskovec, Kevin J Lang, Anirban Dasgupta, and Michael W Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics* 6:1 (2009), pp. 29–123 (cited on page 55).

[LG14] Omer Levy and Yoav Goldberg. Neural Word Embedding as Implicit Matrix Factorization. In *NIPS*, Montreal, Canada, 2014, 2177–2185 (cited on page 35).

[Li+19] Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. DeepGCNs: Can GCNs go as deep as CNNs? In *ICCV*, 2019, pp. 9266–9275 (cited on pages 48, 53).

[LSH13] Ping Li, Gennady Samorodnitsky, and John Hopcroft. Sign Cauchy Projections and Chi-Square Kernel. In *NIPS*, 2013 (cited on page 26).

[Li+23] Yandi Li, Haobo Gao, Yunxuan Gao, Jianxiong Guo, and Weili Wu. A Survey on Influence Maximization: From an ML-Based Combinatorial Optimization. *ACM Trans. Knowl. Discov. Data* 17:9 (2023) (cited on page 72).

[Li+18] Yuchen Li, Ju Fan, Yanhao Wang, and Kian-Lee Tan. Influence Maximization on Social Graphs: A Survey. *IEEE Trans. Knowl. Data Eng.* 30:10 (2018), pp. 1852–1872 (cited on pages 5, 71, 72).

[Lia+18] Defu Lian, Kai Zheng, Vincent W. Zheng, Yong Ge, Longbing Cao, Ivor W. Tsang, and Xing Xie. High-Order Proximity Preserving Information Network Hashing. In *KDD*, 2018, pp. 1744–1753 (cited on page 24).

[LGP21] Jiongqian Liang, Saket Gurukar, and Srinivasan Parthasarathy. MILE: A Multi-Level Framework for Scalable Graph Embedding. In *ICWSM*, 2021, pp. 361–372 (cited on pages 23, 24).

[Lim+15] Néhémy Lim, Florence D'alché-Buc, Cédric Auliac, and George Michailidis. Operator-Valued Kernel-Based Vector Autoregressive Models for Network Inference. *Mach. Learn.* 99:3 (2015), 489–513 (cited on page 88).

[LLY11] Yong Lin, Linyuan Lu, and Shing-Tung Yau. Ricci curvature of graphs. *Tohoku Mathematical Journal, Second Series* 63:4 (2011), pp. 605–627 (cited on page 52).

[LC01] Stan Lipovetsky and Michael Conklin. Analysis of regression in game theory approach. *Appl. Stochastic Models Bus. Ind* 17:4 (2001), pp. 319–330 (cited on page 64).

[Liu+22a] Bin Liu, Dimitrios Papadopoulos, Fragkiskos D. Malliaros, Grigorios Tsoumakas, and Apostolos N Papadopoulos. Multiple similarity drug–target interaction prediction with random walks and matrix factorization. *Briefings in Bioinformatics* 23:5 (2022), bbac353 (cited on pages 3, 4, 31, 40, 43).

[Liu+22b] Bin Liu, Konstantinos Pliakos, Celine Vens, and Grigorios Tsoumakas. Drug-target interaction prediction via an ensemble of weighted nearest neighbors with interaction recovery. *Appl. Intell.* 52:4 (2022), pp. 3705–3727 (cited on page 44).

[LT21] Bin Liu and Grigorios Tsoumakas. Optimizing Area Under the Curve Measures via Matrix Factorization for Predicting Drug-Target Interaction with Multiple Similarities. *arXiv* abs/2105.01545 (2021), pp. 1–14 (cited on pages 43, 44).

[Liu+23a] Chengyi Liu, Wenqi Fan, Yunqing Liu, Jiatong Li, Hang Li, Hui Liu, Jiliang Tang, and Qing Li. Generative Diffusion Models on Graphs: Methods and Applications. In *IJCAI*, 2023, pp. 6702–6711 (cited on page 89).

[Liu+23b] Chuang Liu, Yibing Zhan, Jia Wu, Chang Li, Bo Du, Wenbin Hu, Tongliang Liu, and Dacheng Tao. Graph Pooling for Graph Neural Networks: Progress, Challenges, and Opportunities. In *IJCAI*, 2023, pp. 6712–6722 (cited on page 55).

[Liu+16a] Xinyue Liu, Chara Aggarwal, Yu-Feng Li, Xiaugnan Kong, Xinyuan Sun, and Saket Sathe. Kernelized Matrix Factorization for Collaborative Filtering. In *SDM*, 2016, pp. 378–386 (cited on page 16).

[Liu+23c] Yang Liu, Chuan Zhou, Shirui Pan, Jia Wu, Zhao Li, Hongyang Chen, and Peng Zhang. CurvDrop: A Ricci Curvature Based Approach to Prevent Graph Neural Networks from Over-Smoothing and Over-Squashing. In *The Web Conference*, 2023 (cited on pages 49, 54).

[Liu+15] Yang Liu et al. Topical Word Embeddings. In *AAAI*, 2015, pp. 2418–2424 (cited on page 9).

[Liu+16b] Yong Liu, Min Wu, Chunyan Miao, Peilin Zhao, and Xiao Li Li. Neighborhood regularized logistic matrix factorization for drug-target interaction prediction. *PLoS Comput. Biol.* 12:2 (2016), e1004760 (cited on pages 40, 41, 44).

[LS19] Andrey Y Lokhov and David Saad. Scalable influence estimation without sampling. *CoRR* (2019) (cited on pages 81, 83).

[LL17] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *NeurIPS*, 2017 (cited on pages 62, 64, 66, 68).

[Luo+20] Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. Parameterized Explainer for Graph Neural Network. In *NeurIPS*, 2020 (cited on pages 62, 66).

[Luo+17] Yunan Luo, Xinbin Zhao, Jingtian Zhou, Jinglin Yang, Yanqing Zhang, Wenhua Kuang, Jian Peng, Ligong Chen, and Jianyang Zeng. A network integration approach for drug-target interaction prediction and computational drug repositioning from heterogeneous information. *Nature Communications* 8:1 (2017), pp. 1–13 (cited on pages 40, 44).

[MT21] Yao Ma and Jiliang Tang. *Deep Learning on Graphs*. Cambridge University Press, 2021 (cited on page 7).

[Ma+19] Yao Ma, Suhang Wang, Charu C Aggarwal, and Jiliang Tang. Graph convolutional networks with eigenpooling. In *KDD*, 2019 (cited on page 61).

[Mal15] Fragkiskos D. Malliaros. *Mining Social and Information Networks: Dynamics and Applications*. Theses. Ecole Doctorale de l'Ecole Polytechnique. 2015. URL: https://pastel.hal.science/tel-01245134 (cited on page 2).

[Mal+20] Fragkiskos D. Malliaros, Christos Giatsidis, Apostolos N. Papadopoulos, and Michalis Vazirgiannis. The core decomposition of networks: theory, algorithms and applications. *VLDB J.* 29:1 (2020), pp. 61–92 (cited on pages 2, 77).

[MPV16] Fragkiskos D. Malliaros, Apostolos N. Papadopoulos, and Michalis Vazirgiannis. Core Decomposition in Graphs: Concepts, Algorithms and Applications. In *EDBT*, 2016 (cited on page 2).

[MRV16] Fragkiskos D. Malliaros, Maria-Evgenia G Rossi, and Michalis Vazirgiannis. Locating influential nodes in complex networks. *Scientific Reports* 6 (2016), p. 19307 (cited on pages 2, 77).

[MS15] Fragkiskos D. Malliaros and Konstantinos Skianis. Graph-based term weighting for text categorization. In *ASONAM*, 2015 (cited on page 2).

[MV13a] Fragkiskos D. Malliaros and Michalis Vazirgiannis. Clustering and Community Detection in Directed Networks: A Survey. *Physics Reports* (2013) (cited on pages 2, 9, 56, 60).

[MV13b] Fragkiskos D. Malliaros and Michalis Vazirgiannis. To Stay or Not to Stay: Modeling Engagement Dynamics in Social Graphs. In *CIKM*, 2013 (cited on page 2).

[MV15] Fragkiskos D. Malliaros and Michalis Vazirgiannis. Vulnerability assessment in social networks under cascade-based node departures. *Europhysics Letters* 110:6 (2015), p. 68006 (cited on page 2).

[McC+00] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval* 3:2 (2000), pp. 127–163 (cited on page 53).

[MFD20] Filippo Menczer, Santo Fortunato, and Clayton A Davis. *A first course in network science*. Cambridge University Press, 2020 (cited on page 1).

[MSK20] Diego Mesquita, Amauri H. Souza, and Samuel Kaski. Rethinking Pooling in Graph Neural Networks. In *NeurIPS*, 2020 (cited on page 55).

[Mik+13a] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. In *ICLR*, 2013 (cited on page 8).

[Mik+13b] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and Their Compositionality. In *NIPS*, 2013, pp. 3111–3119 (cited on pages 8, 12).

[Mil+02] Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. Network motifs: simple building blocks of complex networks. *Science* 298:5594 (2002), pp. 824–827 (cited on page 56).

[Min+22] Marco Minici, Federico Cinus, Corrado Monti, Francesco Bonchi, and Giuseppe Manco. Cascade-Based Echo Chamber Detection. In *CIKM*, 2022 (cited on page 71).

[MMV17] Marc Mitri, Fragkiskos D. Malliaros, and Michalis Vazirgiannis. Sensitivity of Community Structure to Network Uncertainty. In *SDM*, ed. by Nitesh V. Chawla and Wei Wang. SIAM, 2017 (cited on page 2).

[MM15] Flaviano Morone and Hernán A. Makse. Influence maximization in complex networks through optimal percolation. *Nature* 524:7563 (2015), pp. 65–68 (cited on page 72).

[Mor+20] Christopher Morris, Nils M Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. *CoRR* (2020) (cited on page 60).

[Nam+12] Galileo Namata, Ben London, Lise Getoor, Bert Huang, and U Edu. Query-driven active surveying for collective classification. In *International Workshop on Mining and Learning with Graphs*, 2012 (cited on page 53).

[Nel+19] Walter Nelson, Marinka Zitnik, Bo Wang, Jure Leskovec, Anna Goldenberg, and Roded Sharan. To embed or not: network embedding as a paradigm in computational biology. *Frontiers in genetics* 10 (2019), p. 381 (cited on page 32).

[New18] Mark Newman. *Networks*. Oxford university press, 2018 (cited on page 1).

[New10] Mark Newman. *Networks: An Introduction*. 2010 (cited on page 22).

[NM18] Duong Nguyen and Fragkiskos D. Malliaros. BiasedWalk: Biased Sampling for Representation Learning on Graphs. In *Big Data*, 2018, pp. 4045–4053 (cited on page 33).

[Ngu+23] Khang Nguyen, Nong Minh Hieu, Vinh Duc Nguyen, Nhat Ho, Stanley Osher, and Tan Minh Nguyen. Revisiting Over-smoothing and Over-squashing Using Ollivier-Ricci Curvature. In *ICML*, 2023 (cited on pages 52, 53).

[OAB18] Rawan S. Olayan, Haitham Ashoor, and Vladimir B. Bajic. DDR: Efficient computational method to predict drug-Target interactions using graph mining and machine learning approaches. *Bioinformatics* 7:34 (2018), pp. 1164–1173 (cited on page 40).

[Oll09] Yann Ollivier. Ricci curvature of Markov chains on metric spaces. *Journal of Functional Analysis* 256:3 (2009), pp. 810–864 (cited on pages 51, 52).

[OS20] Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. In *ICLR*, 2020 (cited on pages 48–50).

[Ou+16] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. Asymmetric Transitivity Preserving Graph Embedding. In *KDD*, 2016, pp. 1105–1114 (cited on pages 8, 16, 17, 21, 24, 25).

[Pal+05] Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature* 435 (2005), p. 814 (cited on page 10).

[PM21] George Panagopoulos and Fragkiskos D. Malliaros. Influence Learning and Maximization. In *ICWE*, 2021, pp. 547–550 (cited on page 72).

[PMV18] George Panagopoulos, Fragkiskos D. Malliaros, and Michalis Vazirgiannis. DiffuGreedy: An Influence Maximization Algorithm Based on Diffusion Cascades. In *Complex Networks*, 2018 (cited on pages 74, 78).

[PMV20] George Panagopoulos, Fragkiskos D. Malliaros, and Michalis Vazirgiannis. Influence Maximization Using Influence and Susceptibility Embeddings. In *ICWSM*, 2020, pp. 511–521 (cited on pages 3, 5, 71, 73, 78).

[PMV22] George Panagopoulos, Fragkiskos D. Malliaros, and Michalis Vazirgiannis. Multi-Task Learning for Influence Estimation and Maximization. *IEEE Trans. Knowl. Data Eng.* 34:9 (2022), pp. 4398–4409 (cited on pages 3, 5, 71, 73, 77).

[Pan+23] George Panagopoulos, Nikolaos Tziortziotis, Michalis Vazirgiannis, and Fragkiskos D. Malliaros. Maximizing Influence with Graph Neural Networks. In *ASONAM*, 2023 (cited on pages 3, 5, 71, 79, 83).

[PRF23] Siddharth Patwardhan, Filippo Radicchi, and Santo Fortunato. Influence maximization: Divide and conquer. *Phys. Rev. E* 107 (5 2023), p. 054306 (cited on page 72).

[Ped+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830 (cited on pages 21, 27, 38).

[PMM18] Sen Pei, Flaviano Morone, and Hernán A Makse. Theories for influencer identification in complex networks. In *Complex Spreading Phenomena in Social Systems*, 2018, pp. 125–148 (cited on pages 73, 74, 78).

[PW10] Ofir Pele and Michael Werman. The Quadratic-Chi Histogram Distance Family. In *ECCV*, 2010, pp. 749–762 (cited on page 24).

[PARS14] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. DeepWalk: Online Learning of Social Representations. In *KDD*, 2014, pp. 701–710 (cited on pages 2, 8, 10, 21, 35).

[PL+21] Léo Pio-Lopez, Alberto Valdeolivas, Laurent Tichit, Élisabeth Remy, and Anaïs Baudot. MultiVERSE: a multiplex and multiplex-heterogeneous network embedding approach. *Scientific Reports* 11:1 (2021), p. 8794 (cited on page 37).

[PVT21] Konstantinos Pliakos, Celine Vens, and Grigorios Tsoumakas. Predicting drug-target interactions with multi-label classification and label partitioning. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 18:4 (2021), pp. 1596–1607 (cited on page 40).

[Pop+19] Phillip E Pope, Soheil Kolouri, Mohammad Rostami, Charles E Martin, and Heiko Hoffmann. Explainability methods for graph convolutional neural networks. In *CVPR*, 2019 (cited on page 62).

[QCa+23] Quentin QCappart, Didier Chételat, Elias B. Khalil, Andrea Lodi, Christopher Morris, and Petar Veličković. Combinatorial Optimization and Reasoning with Graph Neural Networks. *Journal of Machine Learning Research* 24:130 (2023), pp. 1–61 (cited on page 88).

[Qiu+19] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Chi Wang, Kuansan Wang, and Jie Tang. NetSMF: Large-Scale Network Embedding as Sparse Matrix Factorization. In *WWW*, 2019, pp. 1509–1520 (cited on pages 24, 27).

[Qiu+18a] Jiezhong Qiu, Jian Tang, Hao Ma, Yuxiao Dong, Kuansan Wang, and Jie Tang. DeepInf: Social Influence Prediction with Deep Learning. In *KDD*, 2018 (cited on page 77).

[Qiu+18b] Jiezhong Qiu et al. Network Embedding As Matrix Factorization: Unifying DeepWalk, LINE, PTE, and Node2Vec. In *WSDM*, 2018, pp. 459–467 (cited on pages 21, 35).

[RST20] Ekagra Ranjan, Soumya Sanyal, and Partha Talukdar. Asap: Adaptive structure aware pooling for learning hierarchical graph representations. In *AAAI*, 2020 (cited on page 61).

[RSG16] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should I trust you?" Explaining the predictions of any classifier. In *KDD*, 2016 (cited on pages 62, 68).

[Ron+20] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. DropEdge: Towards deep graph convolutional networks on node classification. In *ICLR*, 2020 (cited on pages 48, 50, 51, 53).

[Ros+20] Ryan A. Rossi, Nesreen K. Ahmed, Eunyee Koh, Sungchul Kim, Anup Rao, and Yasin Abbasi-Yadkori. A Structural Graph Representation Learning Framework. In *WSDM*, 2020, 483–491 (cited on page 8).

[RAS21] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. *J. Complex Netw.* 9:2 (2021), pp. 1–22 (cited on page 53).

[Roz+19] Benedek Rozemberczki, Ryan Davies, Rik Sarkar, and Charles Sutton. GEMSEC: Graph Embedding with Self Clustering. In *ASONAM*, 2019, pp. 65–72 (cited on page 21).

[SCT21] Michael Sejr Schlichtkrull, Nicola De Cao, and Ivan Titov. Interpreting Graph Neural Networks for NLP With Differentiable Edge Masking. In *ICLR*, 2021 (cited on page 62).

[SSM97] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Kernel principal component analysis. In *ICANN*, 1997, pp. 583–588 (cited on page 16).

[SSo1] Bernhard Scholkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2001 (cited on page 16).

[Sel+17] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *ICCV*, 2017 (cited on page 62).

[Sen+08] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine* 29:3 (2008), pp. 93–93 (cited on page 53).

[Sha53] Lloyd S Shapley. A value for n-person games. *Contributions to the Theory of Games* 2:28 (1953), pp. 307–317 (cited on pages 62, 64).

[SM00] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach.* 22:8 (2000), pp. 888–905 (cited on page 56).

[SGK17] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning Important Features Through Propagating Activation Differences. In *ICML*, 2017 (cited on page 62).

[Sid+17] Nicholas D. Sidiropoulos, Lieven De Lathauwer, Xiao Fu, Kejun Huang, Evangelos E. Papalexakis, and Christos Faloutsos. Tensor Decomposition for Signal Processing and Machine Learning. *IEEE Transactions on Signal Processing* 65:13 (2017), pp. 3551–3582 (cited on page 32).

[SMV18] Konstantinos Skianis, Fragkiskos D. Malliaros, and Michalis Vazirgiannis. Fusing Document, Collection and Label Graph-based Representations with Word Embeddings for Text Classification. In *NAACL-HLT, TextGraphs Workshop*, 2018 (cited on page 2).

[SJ03] Nathan Srebro and Tommi Jaakkola. Weighted Low-rank Approximations. In *ICML*, 2003, pp. 720–727 (cited on page 17).

[Ste02] Ingo Steinwart. On the Influence of the Kernel on the Consistency of Support Vector Machines. *J. Mach. Learn. Res.* 2 (2002), pp. 67–93 (cited on page 18).

[SK10] Erik Strumbelj and Igor Kononenko. An efficient explanation of individual classifications using game theory. *JMLR* 11 (2010), pp. 1–18 (cited on page 64).

[Sub+20] Indhupriya Subramanian, Srikant Verma, Shiva Kumar, Abhay Jere, and Krishanpal Anamika. Multi-omics Data Integration, Interpretation, and Its Application. *Bioinformatics and Biology Insights* 14 (2020), pp. 1–24 (cited on page 32).

[Szk+20] Damian Szklarczyk, Annika L Gable, Katerina C Nastou, David Lyon, Rebecca Kirsch, Sampo Pyysalo, Nadezhda T Doncheva, Marc Legeay, Tao Fang, Peer Bork, Lars J Jensen, and Christian von Mering. The STRING database in 2021: customizable protein–protein networks, and functional characterization of user-uploaded gene/measurement sets. *Nucleic Acids Research* 49:D1 (2020), pp. D605–D612 (cited on page 37).

[Tan+15] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. LINE: Large-scale Information Network Embedding. In *WWW*, 2015, pp. 1067–1077 (cited on page 21).

[Tan+09] Jie Tang, Jimeng Sun, Chi Wang, and Zi Yang. Social influence analysis in large-scale networks. In *KDD*, 2009 (cited on page 53).

[TSX15] Youze Tang, Yanchen Shi, and Xiaokui Xiao. Influence maximization in near-linear time: A martingale approach. In *SIGMOD*, 2015 (cited on pages 78, 83).

[TSL01] Joshua Tenenbaum, Vin Silva, and John Langford. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science (New York, N.Y.)* 290 (Jan. 2001), pp. 2319–23 (cited on page 8).

[Tia+19] Yu Tian, Long Zhao, Xi Peng, and Dimitris Metaxas. Rethinking Kernel Methods for Node Representation Learning on Graphs. In *NIPS*, 2019, pp. 11686–11697 (cited on page 17).

[TMV16] Antoine J.-P. Tixier, Fragkiskos D. Malliaros, and Michalis Vazirgiannis. A Graph Degeneracy-based Approach to Keyword Extraction. In *EMNLP*, 2016, pp. 1860–1870 (cited on page 2).

[Tix+19] Antoine J.-P. Tixier, Maria-Evgenia G. Rossi, Fragkiskos D. Malliaros, Jesse Read, and Michalis Vazirgiannis. Perturb and combine to identify influential spreaders in real-world networks. In *ASONAM*, 2019 (cited on page 2).

[Top+22] Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. In *ICLR*, 2022 (cited on pages 49–54).

[Tsi+18] Anton Tsitsulin, Davide Mottin, Panagiotis Karras, and Emmanuel Müller. VERSE: Versatile Graph Embeddings from Similarity Measures. In *WWW*, 2018, 539–548 (cited on page 21).

[Tsi+21] Anton Tsitsulin, Marina Munkhoeva, Davide Mottin, Panagiotis Karras, Ivan Oseledets, and Emmanuel Müller. FREDE: Anytime Graph Embeddings. *Proc. VLDB Endow.* 14:6 (2021), 1102–1110 (cited on pages 24, 25, 27).

[Tsi+23] Anton Tsitsulin, John Palowitch, Bryan Perozzi, and Emmanuel MÃ¼ller. Graph Clustering with Graph Neural Networks. *Journal of Machine Learning Research* 24:127 (2023), pp. 1–21 (cited on page 60).

[TPM17] Charalampos E Tsourakakis, Jakub Pachocki, and Michael Mitzenmacher. Scalable motif-aware graph clustering. In *WWW*, 2017 (cited on page 56).

[Vas+17a] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All You Need. In *NIPS*, 2017 (cited on page 48).

[Vas+17b] Sharan Vaswani, Branislav Kveton, Zheng Wen, Mohammad Ghavamzadeh, Laks VS Lakshmanan, and Mark Schmidt. Model-independent online learning for influence maximization. In *ICML*, 2017 (cited on page 73).

[Vay+19] Titouan Vayer, Nicolas Courty, Romain Tavenard, Laetitia Chapel, and Rémi Flamary. Optimal Transport for structured data with application on graphs. In *ICML*, 2019 (cited on page 88).

[Vel+18] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. *International Conference on Learning Representations* (2018) (cited on pages 48, 68).

[Vem01] Santosh Vempala. *The random projection method*. American Mathematical Soc., 2001 (cited on page 25).

[VF+22] Luca Veyrin-Forrer, Ataollah Kamal, Stefan Duffner, Marc Plantevit, and Céline Robardet. In pursuit of the hidden features of GNN's internal representations. *Data Knowl. Eng.* 142 (2022), p. 102097 (cited on page 62).

[Vig+23] Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. DiGress: Discrete Denoising diffusion for graph generation. In *ICLR*, 2023 (cited on page 89).

[Vis+10] S. V. N. Vishwanathan, Nicol N. Schraudolph, Risi Kondor, and Karsten M. Borgwardt. Graph Kernels. *J Mach Learn Res* 11 (2010), pp. 1201–1242 (cited on page 17).

[VL07] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing* 17:4 (2007), pp. 395–416 (cited on pages 47, 56).

[VT20] Minh N. Vu and My T. Thai. PGM-Explainer: Probabilistic Graphical Model Explanations for Graph Neural Networks. In *NeurIPS*, 2020 (cited on pages 62, 66).

[Wan+19] Fangping Wan, Lixiang Hong, An Xiao, Tao Jiang, and Jianyang Zeng. NeoDTI: Neural integration of neighbor information from a heterogeneous network for discovering new drug-target interactions. *Bioinformatics* 35:1 (2019), pp. 104–111 (cited on page 40).

[Wan+14] Bo Wang, Aziz M. Mezlini, Feyyaz Demir, Marc Fiume, Zhuowen Tu, Michael Brudno, Benjamin Haibe-Kains, and Anna Goldenberg. Similarity network fusion for aggregating data types on a genomic scale. *Nature Methods* 11:3 (2014), pp. 333–337 (cited on page 37).

[WCW12] Chi Wang, Wei Chen, and Yajun Wang. Scalable influence maximization for independent cascade model in large-scale social networks. *Data Min. Knowl. Disc.* 25:3 (2012), pp. 545–576 (cited on page 84).

[WCZ16] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural Deep Network Embedding. In *KDD*, 2016, 1225–1234 (cited on page 8).

[Wan+23] Hanchen Wang, Tianfan Fu, Yuanqi Du, Wenhao Gao, et al. Scientific discovery in the age of artificial intelligence. *Nature* 620:7972 (2023), pp. 47–60 (cited on page 89).

[Wan+18] Jingdong Wang, Ting Zhang, Jingkuan Song, Nicu Sebe, and Heng Tao Shen. A Survey on Learning to Hash. *IEEE Trans. Pattern Anal. Mach. Intell.* 40:4 (2018), pp. 769–790 (cited on page 24).

[Wan+17] Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. Community Preserving Network Embedding. In *AAAI*, 2017, pp. 203–209 (cited on pages 16, 21).

[Wil+21] James D Wilson, Melanie Baybay, Rishi Sankar, Paul Stillman, and Abbie M Popa. Analysis of population functional connectivity data via multilayer network embeddings. *Network Science* 9:1 (2021), pp. 99–122 (cited on page 37).

[Wu+19] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying Graph Convolutional Networks. In *ICML*, 2019 (cited on pages 51, 53).

[Wu+21] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu. A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems* 32:1 (2021), pp. 4–24 (cited on page 47).

[Xie+23] Yaochen Xie, Zhao Xu, Jingtun Zhang, Zhengyang Wang, and Shuiwang Ji. Self-Supervised Learning of Graph Neural Networks: A Unified Review. *IEEE Trans. Pattern Anal. Mach. Intell.* 45:2 (2023), pp. 2412–2429 (cited on page 87).

[Xu+19] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How Powerful are Graph Neural Networks? In *ICLR*, 2019 (cited on page 48).

[Yam+08] Yoshihiro Yamanishi, Michihiro Araki, Alex Gutteridge, Wataru Honda, and Minoru Kanehisa. Prediction of drug-target interaction networks from the integration of chemical and genomic spaces. *Bioinformatics* 24:13 (2008), i232–i240 (cited on page 44).

[Yan+19] Dingqi Yang, Paolo Rosso, Bin Li, and Philippe Cudre-Mauroux. NodeSketch: Highly-Efficient Graph Embeddings via Recursive Sketching. In *KDD*, 2019, pp. 1162–1172 (cited on pages 24, 25, 27).

[YL13] Jaewon Yang and Jure Leskovec. Overlapping Community Detection at Scale: A Nonnegative Matrix Factorization Approach. In *WSDM*, 2013, pp. 587–596 (cited on page 10).

[YA09] B. Yener and E. Acar. Unsupervised Multiway Data Analysis: A Literature Survey. *IEEE Transactions on Knowledge amp; Data Engineering* 21:01 (2009), pp. 6–20 (cited on page 32).

[YCP15] Xinyang Yi, Constantine Caramanis, and Eric Price. Binary Embedding: Fundamental Limits and Fast Algorithm. In *ICML*, 2015, pp. 2162–2170 (cited on page 25).

[Yin+19] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. GNNExplainer: Generating explanations for graph neural networks. In *NeurIPS*, 2019 (cited on pages 62, 66).

[Yin+18] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical Graph Representation Learning with Differentiable Pooling. In *NeurIPS*, 2018 (cited on pages 55, 59, 61).

[Yua+23] H. Yuan, H. Yu, S. Gui, and S. Ji. Explainability in Graph Neural Networks: A Taxonomic Survey. *IEEE Transactions on Pattern Analysis amp; Machine Intelligence* 45:05 (2023), pp. 5782–5799 (cited on page 62).

[YJ20] Hao Yuan and Shuiwang Ji. Structpool: Structured graph pooling via conditional random fields. In *ICLR*, 2020 (cited on page 55).

[Yua+20] Hao Yuan, Jiliang Tang, Xia Hu, and Shuiwang Ji. XGNN: Towards Model-Level Explanations of Graph Neural Networks. In *KDD*, 2020 (cited on pages 62, 66).

[Yue+20] Xiang Yue, Zhen Wang, Jingong Huang, Srinivasan Parthasarathy, Soheil Moosavinasab, Yungui Huang, Simon M. Lin, Wen Zhang, and Ping Zhang. Graph embedding on biomedical networks: methods, applications and evaluations. *Bioinformatics* 36 (4 2020), pp. 1241–1251 (cited on page 32).

[Zha+13] Jing Zhang, Biao Liu, Jie Tang, Ting Chen, and Juanzi Li. Social Influence Locality for Modeling Retweeting Behaviors. In *IJCAI*, 2013 (cited on page 77).

[Zha+23] Xuan Zhang, Limei Wang, Jacob Helwig, Youzhi Luo, Cong Fu, et al. Artificial Intelligence for Science in Quantum, Atomistic, and Continuum Systems. *CoRR* (2023) (cited on page 89).

[Zha+18] Z. Zhang, P. Cui, H. Li, X. Wang, and W. Zhu. Billion-Scale Network Embedding with Iterative Random Projection. In *ICDM*, 2018, pp. 787–796 (cited on pages 23–25, 27).

[Zha+20] Zi Chao Zhang, Xiao Fei Zhang, Min Wu, Le Ou-Yang, Xing Ming Zhao, and Xiao Li Li. A graph regularized generalized matrix factorization model for predicting links in biomedical bipartite networks. *Bioinformatics* 36:11 (2020), pp. 3474–3481 (cited on page 44).

[ZA20] Lingxiao Zhao and Leman Akoglu. PairNorm: Tackling oversmoothing in GNNs. In *ICLR*, 2020 (cited on pages 49, 53).

[Zhe+13] Xiaodong Zheng, Hao Ding, Hiroshi Mamitsuka, and Shanfeng Zhu. Collaborative matrix factorization with multiple similarities for predicting drug-Target interactions. In *KDD*, 2013, pp. 1025–1033 (cited on pages 40, 44).

[Zho+15] Chuan Zhou, Peng Zhang, Wenyu Zang, and Li Guo. On the upper bounds of spread for greedy algorithms in social network influence maximization. *IEEE Transactions on Knowledge and Data Engineering* 27:10 (2015), pp. 2770–2783 (cited on page 80).

[Zho+20] Kaixiong Zhou, Xiao Huang, Yuening Li, Daochen Zha, Rui Chen, and Xia Hu. Towards deeper graph neural networks with differentiable group normalization. In *NeurIPS*, 2020 (cited on page 53).

[Zho+23] Zhiyao Zhou, Sheng Zhou, Bochao Mao, Xuanyi Zhou, Jiawei Chen, Qiaoyu Tan, Daochen Zha, Yan Feng, Chun Chen, and Can Wang. OpenGSL: A Comprehensive Benchmark for Graph Structure Learning. *CoRR* (2023) (cited on page 88).

[ZL17] Marinka Zitnik and Jure Leskovec. Predicting multicellular function through multi-layer tissue networks. *Bioinformatics* 33:14 (2017), pp. i190–i198 (cited on page 37).

[Zit+19] Marinka Zitnik, Francis Nguyen, Bo Wang, Jure Leskovec, Anna Goldenberg, and Michael M Hoffman. Machine learning for integrating data in biology and medicine: Principles, practice, and opportunities. *Information Fusion* 50 (2019), pp. 71–91 (cited on pages 32, 37, 38).

TITRE: A Machine Learning Tour in Network Science

MOTS CLÉS: Apprentissage automatique de graphes, apprentissage de représentation de graphe, réseaux de neurones en graphes, fouille de graphes, science des réseaux

RÉSUMÉ: Les graphes, également appelés réseaux, sont des structures de données largement utilisées pour modéliser des systèmes complexes dans divers domaines, des sciences sociales à la biologie et à l'ingénierie. Leur force réside dans leur capacité à représenter les relations entre entités, telles que les amitiés dans les réseaux sociaux ou les interactions protéines dans les réseaux biologiques. En plus de leurs capacités de modélisation, les graphes offrent un cadre mathématique qui sert à analyser, comprendre et faire des prédictions à partir d'ensembles de données du monde réel. Ce manuscrit HDR présente une partie de mes contributions de recherche dans le domaine de l'apprentissage, des représentations des graphes et de ses applications à la science des réseaux. Il présente les travaux menés après avoir rejoint CentraleSupélec, Université Paris-Saclay en 2017. La première partie du manuscrit analyse les techniques de plonger les nœuds en préservant la structure qui utilisent les marches aléatoires. La deuxième partie aborde le défi du développement des modèles d'apprentissage de représentation pour les graphes multicouches et hétérogènes, en soulignant les applications issues du domaine de la biologie computationnelle. La troisième partie se focalise sur la conception de modèles des reseaux de neurones en graphes expressifs et explicables. Finalement, la dernière partie étudie l'application de l'apprentissage de la représentation de graphe afin d'aborder les problèmes de l'apprentissage et de la maximisation de l'influence sociale dans des réseaux complexes.

TITLE: A Machine Learning Tour in Network Science

KEYWORDS: Graph machine learning, graph representation learning, graph neural networks, graph mining, network science

ABSTRACT: Graphs, also known as networks, are widely used data structures for modeling complex systems in various fields, from the social sciences to biology and engineering. The strength lies in their ability to represent relationships between entities, such as friendships in social networks or protein interactions in biological networks. In addition to their modeling capabilities, graphs offer a mathematical framework to analyze, understand, and make predictions from real-world datasets. This HDR manuscript presents part of my research contributions to the field of graph representation learning and its applications in network science, focusing on the work conducted after joining CentraleSupélec, Université Paris-Saclay in 2017. The first part of the manuscript explores structure-preserving node embedding techniques that leverage random walks. The second part addresses the challenge of developing graph representation learning models for multilayer and heterogeneous graphs, with a specific focus on applications arising from the domain of computational biology. The third part delves into the design of expressive and explainable graph neural network models. Finally, the last part investigates the application of graph representation learning to tackle the well-studied problems of social influence learning and maximization in complex networks.