Graph Learning: A Survey

Feng Xia, *Senior Member, IEEE*, Ke Sun, Shuo Yu, *Member, IEEE*, Abdul Aziz, Liangtian Wan, *Member, IEEE*, Shirui Pan, and Huan Liu, *Fellow, IEEE*

***Abstract*—Graphs are widely used as a popular representation of the network structure of connected data. Graph data can be found in a broad spectrum of application domains such as social systems, ecosystems, biological networks, knowledge graphs, and information systems. With the continuous penetration of artificial intelligence technologies, graph learning (i.e., machine learning on graphs) is gaining attention from both researchers and practitioners. Graph learning proves effective for many tasks, such as classification, link prediction, and matching. Generally, graph learning methods extract relevant features of graphs by taking advantage of machine learning algorithms. In this survey, we present a comprehensive overview on the state-of-the-art of graph learning. Special attention is paid to four categories of existing graph learning methods, including graph signal processing, matrix factorization, random walk, and deep learning. Major models and algorithms under these categories are reviewed, respectively. We examine graph learning applications in areas such as text, images, science, knowledge graphs, and combinatorial optimization. In addition, we discuss several promising research directions in this field.**

***Impact Statement*—Real-world intelligent systems generally rely on machine learning algorithms handling data of various types. Despite their ubiquity, graph data have imposed unprecedented challenges to machine learning due to their inherent complexity. Unlike text, audio and images, graph data are embedded in an irregular domain, making some essential operations of existing machine learning algorithms inapplicable. Many graph learning models and algorithms have been developed to tackle these challenges. This article presents a systematic review of the state-of-the-art graph learning approaches as well as their potential applications. The article serves multiple purposes. First, it acts as a quick reference to graph learning for researchers and practitioners in different areas such as social computing, information retrieval, computer vision, bioinformatics, economics, and e-commence. Second, it presents insights into open areas of research in the field. Third, it aims to stimulate new research ideas and more interests in graph learning.**

***Index Terms*—Deep learning, graph data, graph learning, graph neural networks (GNNs), machine learning, network embedding, network representation learning (NRL).**

Manuscript received January 14, 2021; revised March 21, 2021; accepted March 28, 2021. Date of publication April 27, 2021; date of current version August 20, 2021. This paper was recommended for publication by Associate Editor Yew Soon Ong. *(Corresponding author: Feng Xia.)*

Feng Xia is with the School of Engineering, IT, and Physical Sciences, Federation University Australia, Ballarat, VIC 3353, Australia (e-mail: f.xia@ieee.org).

Ke Sun, Shuo Yu, Abdul Aziz, and Liangtian Wan are with the School of Software, Dalian University of Technology, Dalian 116620, China (e-mail: kern.sun@outlook.com; y_shuo@outlook.com; ciit.abdulaziz@gmail.com; wan.liangtian.2015@ieee.org).

Shirui Pan is with the Faculty of Information Technology, Monash University, Melbourne, VIC 3800, Australia (e-mail: shirui.pan@monash.edu).

Huan Liu is with the School of Computing, Informatics, and Decision Systems Engineering, Arizona State University, Tempe, AZ 85281 USA (e-mail: huanliu@asu.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TAI.2021.3076021.

Digital Object Identifier 10.1109/TAI.2021.3076021

I. Introduction

GRAPHS, also referred to as networks, can be extracted from various real-world relations among abundant entities. Some common graphs have been widely used to formulate different relationships, such as social networks, biological networks, patent networks, traffic networks, citation networks, and communication networks [1]–[3]. A graph is often defined by two sets, i.e., vertex set and edge set. Vertices represent entities in graph, whereas edges represent relationships between those entities. Graph learning has attracted considerable attention because of its wide applications in the real world, such as data mining and knowledge discovery. Graph learning methods have gained increasing popularity for capturing complex relationships, as graphs exploit essential and relevant relations among vertices [4], [5]. For example, in microblog networks, the spread trajectory of rumors can be tracked by detecting information cascades. In biological networks, new treatments for difficult diseases can be discovered by inferring protein interactions. In traffic networks, human mobility patterns can be predicted by analyzing the co-occurrence phenomenon with different timestamps [6]. Efficient analysis of these networks massively depends on the way how networks are represented.

A. What is Graph Learning?

Generally speaking, graph learning refers to machine learning on graphs. Graph learning methods map the features of a graph to feature vectors with the same dimensions in the embedding space. A graph learning model or algorithm directly converts the graph data into the output of the graph learning architecture without projecting the graph into a low dimensional space. Most graph learning methods are based on or generalized from deep learning techniques, because deep learning techniques can encode and represent graph data into vectors. The output vectors of graph learning are in continuous space. The target of graph learning is to extract the desired features of a graph. Thus, the representation of a graph can be easily used by downstream tasks such as node classification and link prediction without an explicit embedding process. Consequently, graph learning is a more powerful and meaningful technique for graph analysis.

In this survey article, we try to examine machine learning methods on graphs in a comprehensive manner. As shown in Fig. 1, we focus on existing methods that fall into the following four categories: graph signal processing (GSP)-based methods, matrix factorization-based methods, random walk-based methods, and deep learning-based methods. Roughly speaking,

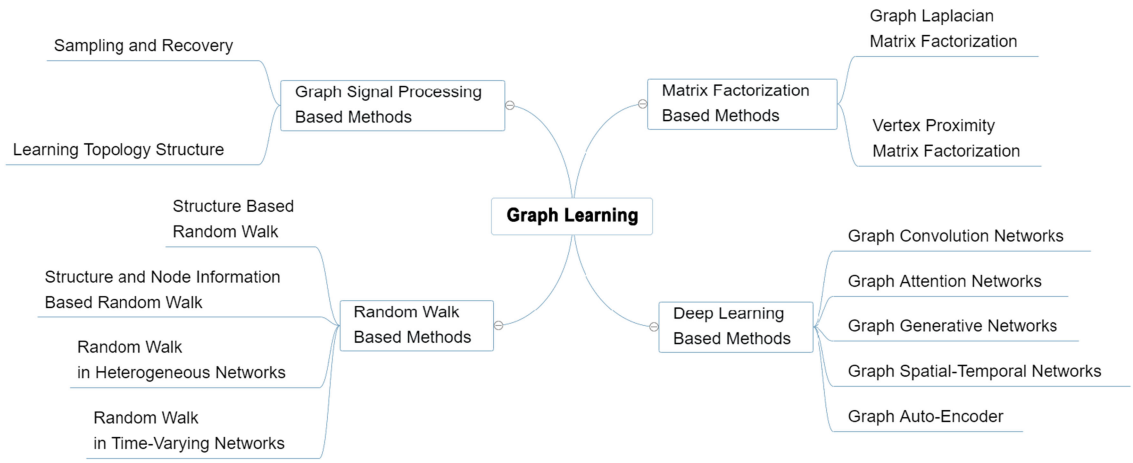


Fig. 1. Categorization of graph learning.

GSP deals with sampling and recovery of graph, and learning topology structure from data. Matrix factorization can be divided into graph Laplacian matrix factorization and vertex proximity matrix factorization. Random walk-based methods include structure-based random walk, structure and node information based random walk, random walk in heterogeneous networks, and random walk in time-varying networks. Deep learning-based methods include graph convolutional networks, graph attention networks, graph auto-encoder, graph generative networks, and graph spatial-temporal networks. Basically, the model architectures of these methods/techniques differ from each other. This article presents an extensive review of the state-of-the-art graph learning techniques.

Traditionally, researchers adopt an adjacency matrix to represent a graph, which can only capture the relationship between two adjacent vertices. However, many complex and irregular structures cannot be captured by this simple representation. When we analyze large-scale networks, traditional methods are computationally expensive and hard to be implemented in real-world applications. Therefore, effective representation of these networks is a paramount problem to solve [4]. Network representation learning (NRL) proposed in recent years can learn latent features of network vertices with low dimensional representation [7]–[9]. When the new representation has been learned, previous machine learning methods can be employed for analyzing the graph data as well as discovering relationships hidden in the data.

When complex networks are embedded into a latent, low dimensional space, the structural information, and vertex attributes can be preserved [4]. Thus, the vertices of networks can be represented by low dimensional vectors. These vectors can be regarded as the features of input in previous machine learning methods. Graph learning methods pave the way for graph analysis in the new representation space, and many graph analytical tasks, such as link prediction, recommendation, and classification, can be solved efficiently [10], [11]. Graphical network representation sheds light on various aspects of social life, such as communication patterns, community structure, and information diffusion [12], [13]. According to the attributes of vertices, edges, and subgraph, graph learning tasks can be divided into three categories, which are vertices-based, edges-based, and subgraph-based, respectively. The relationships among vertices in a graph can be exploited for, e.g., classification, risk identification, clustering, and community detection [14]. By judging the presence of edges between two vertices in graphs, we can perform recommendation and knowledge reasoning, for instance. Based on the classification of subgraphs [15], the graph can be used for, e.g., polymer classification, 3-D visual classification, etc. For GSP, it is significant to design suitable graph sampling methods to preserve the features of the original graph, which aims at recovering the original graph efficiently [16]. Graph recovery methods can be used for constructing the original graph in the presence of incomplete data [17]. Afterward, graph learning can be exploited to learn the topology structure from graph data. In summary, graph learning can be used to tackle the following challenges, which are difficult to solve by using traditional graph analysis methods [18].

1) *Irregular domains:* Data collected by traditional sensors have a clear grid structure. However, graphs lie in an irregular domain (i.e., non-Euclidean space). In contrast to regular domain (i.e., Euclidean space), data in non-Euclidean space are not ordered regularly. Distance is hence difficult to be defined. As a result, basic methods based on traditional machine learning and signal processing cannot be directly generalized to graphs.
2) *Heterogeneous networks:* In many cases, networks involved in the traditional graph analysis algorithms are homogeneous. The appropriate modeling methods only consider the direct connection of the network and strip other irrelevant information, which significantly simplifies the processing. However, it is prone to cause information loss. In the real world, the edges among vertices and the types of vertices are usually diverse, such as in the academic network shown in Fig. 2. Thus it is not easy to discover potential value from heterogeneous information networks with abundant vertices and edges.
3) *Distributed algorithms:* In big social networks, there are often millions of vertices and edges [19]. Centralized

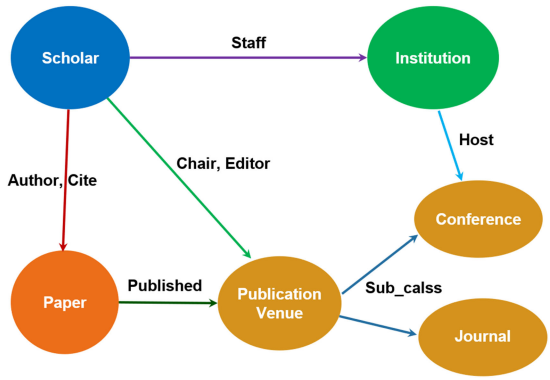


Fig. 2. Heterogeneous academic network [28].

algorithms cannot handle this since the computational complexity of these algorithms would significantly increase with the growth of vertex number. The design of distributed algorithms for dealing with big networks is a critical problem yet to be solved [20]. One major benefit of distributed algorithms is that the algorithms can be executed in multiple CPUs or GPUs simultaneously, and hence, the running time can be reduced significantly.

B. Related Surveys

There are several surveys that are partially related to the scope of this article. Unlike these surveys, we aim to provide a comprehensive overview of graph learning methods, with a focus on four specific categories. In particular, GSP is introduced as one approach for graph learning, which is not covered by other surveys.

Goyal and Ferrara [21] summarized graph embedding methods, such as matrix factorization, random walk, and their applications in graph analysis. Cai *et al.* [22] reviewed graph embedding methods based on problem settings and embedding techniques. Zhang *et al.* [4] summarized NRL methods based on two categories, i.e., unsupervised NRL and semisupervised NRL, and discussed their applications. Nickel *et al.* [23] introduced knowledge extraction methods from two aspects: latent feature models and graph-based models. Akoglu *et al.* [24] reviewed state-of-the-art techniques for event detection in data represented as graphs, and their applications in the real world. Zhang *et al.* [18] summarized deep learning-based methods for graphs, such as graph neural networks (GNNs), graph convolutional networks (GCNs), and graph auto-encoders (GAEs). Wu *et al.* [25] reviewed state-of-the-art GNN methods and discussed their applications in different fields. Ortega *et al.* [26] introduced GSP techniques for representation, sampling and learning, and discussed their applications. Huang *et al.* [27] examined the applications of GSP in functional brain imaging and addressed the problem of how to perform brain network analysis from signal processing perspective.

In summary, none of the existing surveys provides a comprehensive overview of graph learning. They only cover some parts of graph learning, such as network embedding and deep learning-based network representation. The NRL and/or GNN-based surveys do not cover the GSP techniques. In contrast, we review GSP techniques in the context of graph learning, as it is an important approach for GNNs. Specifically, this survey article integrates state-of-the-art machine learning techniques for graph data, gives a general description of graph learning, and discusses its applications in various domains.

C. Contributions and Organization

The contributions of this article can be summarized as follows.

1) *A comprehensive overview of state-of-the-art graph learning methods:* We present an integral introduction to graph learning methods, including, e.g., technical sketches, application scenarios, and potential research directions.
2) *Taxonomy of graph learning:* We give a technical classification of mainstream graph learning methods from the perspective of theoretical models. Technical descriptions are provided wherever appropriate to improve understanding of the taxonomy.
3) *Insights into future directions in graph learning:* Besides qualitative analysis of existing methods, we shed light on potential research directions in the field of graph learning through summarizing several open issues and relevant challenges.

The rest of this article is organized as follows. An overview of graph learning approaches containing GSP-based methods, matrix factorization-based methods, random walk-based methods, and deep learning-based methods is provided in Section II. The applications of graph learning are examined in Section III. Some future directions as well as challenges are discussed in Section IV. Finally, Section V concludes the article.

II. Graph Learning Models and Algorithms

The feature vectors that represent various categorical attributes are viewed as the input in previous machine learning methods. However, the mapping from the input feature vectors to the output prediction results need to be handled by graph learning [21]. Deep learning has been regarded as one of the most successful techniques in artificial intelligence [29], [30]. Extracting complex patterns by exploiting deep learning from a massive amount of irregular data has been found very useful in various fields, such as pattern recognition and image processing. Consequently, how to utilize deep learning techniques to extract patterns from complex graphs has attracted lots of attention. Deep learning on graphs, such as GNNs, GCNs, and GAEs, has been recognized as a powerful technique for graph analysis [18]. Besides, GSP has also been proposed to deal with graph analysis [26]. One of the most typical scenarios is that a set of values reside on a set of vertices, and these vertices are connected by edges [31]. Graph signals can be adopted to model various phenomena in real world. For example, in social networks, users in Facebook can be viewed as vertices, and their friendships can be modeled as edges. The number of followers of each vertex is marked in this social network. Based on this assumption, many techniques (e.g., convolution, filter, wavelet,

TABLE I
DEFINITIONS OF ABBREVIATIONS

Abbreviation	Definition
PCA	Principal component analysis
NRL	Network representation learning
LSTM	Long short-term memory (networks)
GSP	Graph signal processing
GNN	Graph neural network
GMRF	Gauss markov random field
GCN	Graph convolutional network
GAT	Graph attention network
GAN	Generative adversarial network
GAE	Graph auto-encoder
ASP	Algebraic signal processing
RNN	Recurrent neural network
CNN	Convolutional neural network

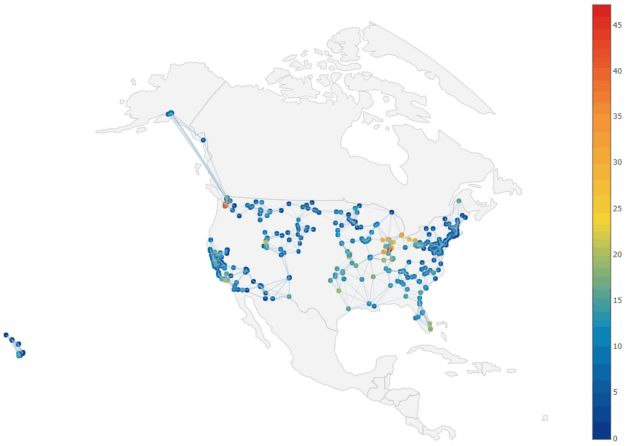


Fig. 3. Measurements of PM2.5 from different sensors on July 5, 2014 (data source: https://www.epa.gov/).

etc.) in classical signal processing can be employed for GSP with suitable modifications [26].

In this section, we review graph learning models and algorithms under four categories as mentioned before—namely, GSP-based methods, matrix factorization-based methods, random walk-based methods, and deep learning-based methods. In Table I, we list the abbreviations used in this article.

A. Graph Signal Processing

Signal processing is a traditional subject that processes signals defined in regular data domain. In recent years, researchers extend concepts of traditional signal processing into graphs. Classical signal processing techniques and tools such as Fourier transform and filtering can be used to analyze graphs. In general, graphs are a kind of irregular data, which are hard to handle directly. As a complement to learning methods based on structures and models, GSP provides a new perspective of spectral analysis of graphs. Derived from signal processing, GSP can give an explanation of graph property consisting of connectivity, similarity, etc. Fig. 3 gives a simple example of graph signals at a certain time point, which is defined as observed values. In a graph, the abovementioned observed values can be regarded as graph signals. Each node is then mapped to the real number field in GSP. The main task of GSP is to expand signal processing approaches to mine implicit information in graphs.

1) Representation on Graphs: A meaningful representation of graphs has contributed a lot to the rapid growth of graph learning. There are two main models of GSP, i.e., adjacency matrix-based GSP [31] and Laplacian-based GSP [32]. An adjacency matrix-based GSP comes from algebraic signal processing (ASP) [33], which interprets linear signal processing from algebraic theory. Linear signal processing contains signals, filters, signal transformation, etc. It can be applied in both continuous and discrete time domains. The basic assumption of linear algebra is extended to the algebra space in ASP. By selecting signal model appropriately, ASP can obtain different instances in linear signal processing. In adjacency matrix-based GSP, the signal model is generated from a shift. Similar to traditional signal processing, a shift in GSP is a filter in graph domain [31], [34], [35]. GSP usually defines graph signal models using adjacency matrices as shifts. Signals of a graph are normally defined at vertices.

Laplacian-based GSP originates from spectral graph theory. High dimensional data are transferred into a low dimensional space generated by a part of the Laplacian basis [36]. Some researchers exploited sensor networks [37] to achieve distributed processing of graph signals. Other researchers solved the problem globally under the assumption that the graph is smooth. Unlike adjacency matrix-based GSP, Laplacian matrix is symmetric with real and nonnegative edge weights, which is used to index undirected graphs.

Although the models use different matrices as basic shifts, most of the notions in GSP are derived from signal processing. Notions with different definitions in these models may have similar meanings. All of them correspond to concepts in signal processing. Signals in GSP are values defined on graphs, and they are usually written as a vector, $\boldsymbol{s} = [s_0, s_1, \ldots, s_{N-1}] \in \mathbb{C}^N$. N is the number of vertices, and each element in the vector represents the value on a vertex. Some studies [26] allow complex-value signals, even though most applications are based on real-value signals.

In the context of adjacency matrix-based GSP, a graph can be represented as a triple $G(V, E, \boldsymbol{W})$, where V is the vertex set, E is the edge set, and $\mathbf{W}$ is the adjacency matrix. With the definition of graphs, we can also define degree matrix $\boldsymbol{D}_{ii} = \boldsymbol{d}_i$, where $\boldsymbol{D}$ is a diagonal matrix, and $\boldsymbol{d}_i$ is the degree of vertex i. Graph Laplacian is defined as $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}$, and normalized Laplacian is defined as $\boldsymbol{L}_{norm} = \boldsymbol{D}^{-1/2}\boldsymbol{L}\boldsymbol{D}^{-1/2}$. Filters in signal processing can be seen as a function that amplifies or reduces relevant frequencies, eliminating irrelevant ones. Matrix multiplication in linear space equals to scale changing, which is identical with filter operation in frequency domain. It is obvious that we can use matrix multiplication as a filter in GSP, which is written as $\boldsymbol{s}_{\text{out}} = \boldsymbol{H}\boldsymbol{s}_{\text{in}}$, where $\boldsymbol{H}$ stands for a filter.

Shift is an important concept to describe variation in signal, and time-invariant signals are used frequently [31]. In fact, there are different choices of shifts in GSP. Adjacency matrix-based GSP uses $\boldsymbol{A}$ as shift. Laplacian-based GSP uses $\mathbf{L}$ [32], and some researchers also use other matrices [38]. By following time invariance in traditional signal processing, shift invariance

is defined in GSP. If filters are commutative with shift, they are shift-invariant, which can be written as $\boldsymbol{AH} = \boldsymbol{HA}$. It is proved that shift-invariant filter can be represented by the shift. The properties of shift are vital, and they determine the fashion of other definitions such as Fourier transform and frequency.

In adjacency matrix-based GSP, eigenvalue decomposition of shift $\boldsymbol{A}$ is $\boldsymbol{A} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{-1}$. $\boldsymbol{V}$ is the matrix of eigenvectors $[\boldsymbol{v}_0, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_{N-1}]$ and

$$\boldsymbol{\Lambda} = \begin{bmatrix} \boldsymbol{\lambda}_0 & & \\ & \ddots & \\ & & \boldsymbol{\lambda}_{N-1} \end{bmatrix}$$

is a diagonal matrix of eigenvalues. The Fourier transform matrix is the inverse of $\boldsymbol{V}$, i.e., $\boldsymbol{F} = \boldsymbol{V}^{-1}$. Frequency of shift is defined as total variation, which states the difference after shift

$$TV_G = ||\boldsymbol{v}_k - \frac{1}{\lambda_{\max}}\boldsymbol{A}\boldsymbol{v}_k||_1$$

where $\frac{1}{\lambda_{\max}}$ is a normalized factor of matrix. It means that the frequencies of eigenvalue far away from the largest eigenvalues on complex plane are large. A large frequency means that signals are changed with a large scale after shift filtering. The differences between minimum and maximum $\boldsymbol{\lambda}$ can be seen in Fig. 4. Generally, the total variation tends to be relatively low with larger frequency, and vice versa. Eigenvectors of larger eigenvalues can be used to construct low-frequency filters, which capture fundamental characteristics, and smaller ones can be employed to capture the variation among neighbor nodes.

For topology learning problems, we can distinguish the corresponding solutions depending on known information. When topology information is partly known, we can use the known information to infer the whole graph. In case the topology information is unknown while we still can observe the signals on the graph, the topology structure has to be inferred from the signals. The former one is often solved as a sampling and recovery problem, and blind topology inference is also known as graph topology (or structure) learning.

2) Sampling and Recovery: Sampling is not a new concept defined in GSP. In conventional signal processing, we normally need to reconstruct original signals with the least samples and retain all information of original signals for a sampling problem. Few samples lead to the lack of information and more samples need more space to store. The well-known Nyquist–Shannon sampling theorem gives the sufficient condition of perfect recovery of signals in time domain.

Researchers have migrated the sampling theories into GSP to study the sampling problem on graphs. As the volume of data is large in some real-world applications such as sensor networks and social networks, sampling less and recovering better are vital for GSP. In fact, most algorithms and frameworks solving sampling problems require that the graph models correlations within signals observed on it [39]. The sampling problem can be defined as reconstructing signals from samples on a subset of vertices, and signals in it are usually band-limited. Nyquist–Shannon sampling theorem was extended to graph signals in [40]. Based on the normalized Laplacian matrix, sampling theorem and

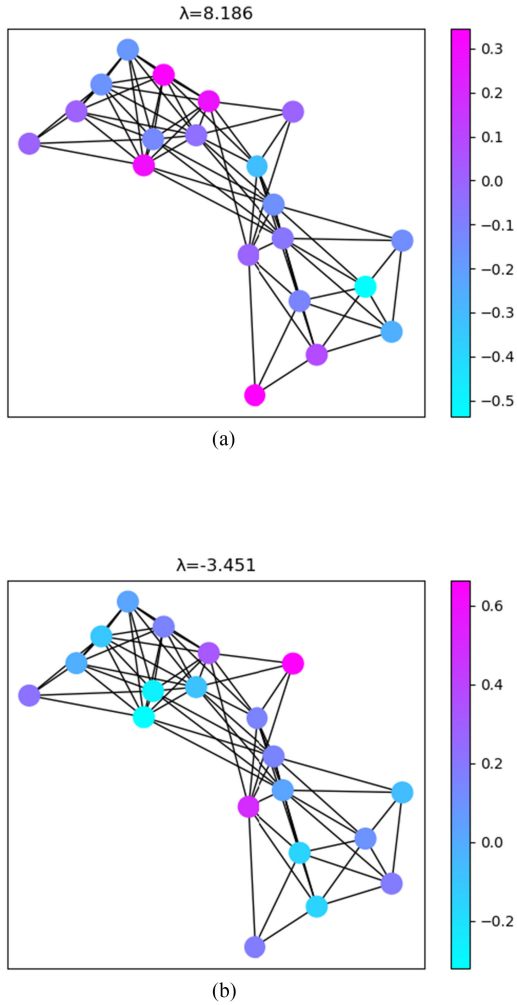


Fig. 4. Illustration of difference between minimum and maximum frequencies. (a) The maximum frequency and (b) The minimum frequency.

cutoff frequency are defined for GSP. Moreover, the authors provided a method for computing cutoff frequency from a given sampling set and a method for choosing sampling set for a given bandwidth. It should be noted that the sampling theorem proposed therein is merely applied to undirected graph. As Laplacian matrix represents undirected graphs only, sampling theory for directed graph adopts adjacent matrix. An optimal operator with a guarantee for perfect recovery was proposed in [35], and it is robust to noise for general graphs.

One of the explicit distinctions between classical signal processing and GSP is that signals of the former fall in regular domain while the latter falls in irregular domain. For sampling and recovery problems, classical signal processing samples successive signals and can recover successive signals from samplings. GSP samples a discrete sequence, and recovers the original sequences from samplings. By following this order, the solution is generally separated into two parts, i.e., finding sampling vertex sets and reconstructing original signals based on various models.

When the size of the dataset is small, we can handle the signal and shift directly. However, for a large-scale dataset,

some algorithms require matrix decomposition to obtain frequencies and save eigenvalues in the procedure, which are almost impossible to realize. As a simple technique applicable to large-scale datasets, a random method can also be used in sampling. Puy *et al.* [41] proposed two sample strategies: a non-adaptive one depending on a parameter and an adaptive random sampling strategy. By relaxing the optimized constraint, they extended random sampling to large scale graphs. Another common strategy is greedy sampling. For example, Shomorony and Avestimehr [42] proposed an efficient method based on linear algebraic conditions that can exactly compute cutoff frequency. Chamon and Ribeiro [43] provided near-optimal guarantee for greedy sampling, which guarantees the performance of greedy sampling in the worst cases.

All of the sampling strategies mentioned above can be categorized as selecting sampling, where signals are observed on a subset of vertices [43]. Besides selecting sampling, there exists a type of sampling called aggregation sampling [44], which uses observations taken at a single vertex as input, containing a sequential applications of graph shift operator.

Similar to classical signal processing, the reconstruction task on graphs can also be interpreted as data interpolation problem [45]. By projecting the samples on a proper signal space, researchers obtain interpolated signals. Least squares reconstruction is an available method in practice. Gadde and Ortega [46] defined a generative model for signal recovery derived from a pairwise Gaussian random field (GRF) and a covariance matrix on graphs. Under sampling theorem, the reconstruction of graph signals can be viewed as the maximum posterior inference of GRF with low-rank approximation. Wang *et al.* [47] aimed at achieving the distributed reconstruction of time-varying band limited signal, where the distributed least squares reconstruction (DLSR) was proposed to recover the signals iteratively. DLSR can track time-varying signals and achieve perfect reconstruction. Di Lorenzo *et al.* [48] proposed a linear mean squares (LMS) strategy for adaptive estimation. LMS enables online reconstruction and tracking from the observation on a subset of vertices. It also allows the subset to vary over time. Moreover, a sparse online estimation was proposed to solve the problems with unknown bandwidth.

Another common technique for recovering original signals is smoothness. Smoothness is used for inferring missing values in graph signals with low frequencies. Wang *et al.* [17] defined the concept of local set. Based on the definition of graph signals, two iterative methods were proposed to recover band limited signals on graphs. Besides, Romero *et al.* [49] advocated kernel regression as a framework for GSP modeling and reconstruction. For parameter selection in estimators, two multikernel methods were proposed to solve a single optimization problem as well. In addition, some researchers investigated different recovery problems with compressed sensing [50].

In addition, there exists some researches on sampling of different kinds of signals such as smooth graph signals, piece-wise constant signals and piecewise smooth signals [51]. Chen *et al.* [51] gave a uniform framework to analyze graph signals. The reconstruction of a known graph signal was studied in [52], where the signal is sparse, which means only a few vertices are nonzeros. Three kinds of reconstruction schemes corresponding to various seeding patterns were examined. By analyzing single simultaneous injection, single successive value injection, and multiple successive simultaneous injections, the conditions for perfect reconstruction on any vertices were derived.

3) Learning Topology Structure From Data: In most application scenes, graphs are constructed according to connections of entity correlations. For example, in sensor networks, the correlations between sensors are often consistent with geographic distance. Edges in social networks are defined as relations such as friends or colleagues [53]. In biochemical networks, edges are generated by interactions. Although GSP is an efficient framework for solving problems on graphs such as sampling, reconstruction, and detection, there lacks a step to extract relations from datasets. Connections exist in many datasets without explicit records. Fortunately, they can be inferred in many ways.

As a result, researchers want to learn complete graphs from datasets. The problem of learning graph from a dataset is stated as estimating graph Laplacian, or graph topology [54]. Generally, they require the graph to satisfy some properties, such as sparsity and smoothness. Smoothness is a widespread assumption in networks generated from datasets. Therefore, it is usually used to constrain observed signals and provide a rational guarantee for graph signals. Researchers have applied it to graph topology learning. The intuition behind smoothness based algorithms is that most signals on graph are stationary, and the result filtered by shift tends to be the lowest frequency. Dong *et al.* [55] adopted a factor analysis model for graph signals, and also imposed a Gaussian prior on latent variables to obtain a principal component analysis (PCA) like representation. Kalofolias [56] formulated the objective as a weighted l_1 problem and designed a general framework to solve it.

Gauss Markov random field (GMRF) is also a widely used theory for graph topology learning in GSP [54], [57], [58]. The models of GRMF-based graph topology learning select graphs that are more likely to generate signals which are similar to the ones generated by GMRF. Egilmez *et al.* [54] formulated the problem as a maximum posterior parameter estimation of GMRF, and the graph Laplacian is a precision matrix. Pavez and Ortega [57] also formulated the problem as a precision matrix estimation, and the rows and columns are updated iteratively by optimizing a quadratic problem. Both of them restrict the result matrix, which should be Laplacian. In [58], Pavez *et al.* chose a two steps framework to find the structure of the underlying graph. First, a graph topology inference step is employed to select a proper topology. Then, a generalized graph Laplacian is estimated. An error bound of Laplacian estimation is computed. In the next step, the error bound can be utilized to obtain a matrix in a specific form as the precision matrix estimation. It is one of the first work that suggests adjusting the model to obtain a graph satisfying the requirement of various problems.

Diffusion is also a relevant model that can be exploited to solve the topology interfering problem [39], [59]–[61]. Diffusion refers to that the node continuously influences its neighborhoods. In graphs, nodes with larger values will have higher

influence on their neighborhood nodes. Using a few components to represent signals will help to find the main factors of signal formation. The models of diffusion are often under the assumption of independent identically distributed signals. Pasdeloup *et al.* [59] gave the concept of valid graphs to explain signals and assumed that the signals are observed after diffusion. Segarra *et al.* [60] agreed that there exists a diffusion process in the shift, and the signals can be observed. The signals in [61] were explained as a linear combination of a few components.

For time series recorded in data, researchers tried to construct time-sequential networks. For instance, Mei and Moura [62] proposed a methodology to estimate graphs, which considers both time and space dependencies and models them by autoregressive process. Segarra *et al.* [63] proposed a method that can be seen as an extension of graph learning. The aim of the article was to solve the problem of joint identification of a graph filter and its input signal.

For recovery methods, a well-known partial inference problem is recommendation [45], [64], [65]. The typical algorithm used in recommendation is collaborative filtering (CF) [66]. Given the observed ratings in a matrix, the objective of CF is to estimate the full rating matrix. Huang *et al.* [65] demonstrated that CF could be viewed as a specific band-stop graph filter on networks representing correlations between users and items. Furthermore, linear latent factor methods can also be modeled as band limited interpretation problem.

4) Discussion: GSP algorithms have strict limitations on experimental data, thus leading to less real-world applications. Moreover, GSP algorithms require the input data to be exactly the whole graph, which means that part of graph data cannot be the input. Therefore, the computational complexity of this kind of methods could be significantly high. In comparison with other kinds of graph learning methods, the scalability of GSP algorithms is relatively poor.

B. Matrix Factorization-Based Methods

Matrix factorization is a method of simplifying a matrix into its components. These components have a lower dimension and could be used to represent the original information of a network, such as relationships among nodes. Matrix factorization-based graph learning methods adopt a matrix to represent graph characteristics like vertex pairwise similarity, and the vertex embedding can be achieved by factorizing this matrix [67]. Early graph learning approaches usually utilized matrix factorization-based methods to solve the graph embedding problem. The input of matrix factorization is the nonrelational high dimensional data feature represented as a graph. The output of matrix factorization is a set of vertex embedding. If the input data lies in a low dimensional manifold, the graph learning for embedding can be treated as a dimension-reduced problem that preserves the structure information. There are mainly two types of matrix factorization-based graph learning. One is graph Laplacian matrix factorization, and the other is vertex proximity matrix factorization.

1) Graph Laplacian Matrix Factorization: The preserved graph characteristics can be expressed as pairwise vertex similarities. Generally, there are two kinds of graph Laplacian matrix factorization, i.e., transductive and inductive matrix factorization. The former only embeds the vertices contained in the training set, and the latter can embed the vertices that are not contained in the training set. The general framework has been designed in [68], and the graph Laplacian matrix factorization based graph learning methods have been summarized in [69]. The Euclidean distance between two feature vectors is directly adopted in the initial metric multidimensional scaling (MDS) [70] to find the optimal embedding. The neighborhoods of vertices are not considered in the MDS, i.e., any pair of training instances are considered as connected. The data feature is extracted by constructing a k nearest neighbor graph, and the subsequent studies [67], [71]–[73] tackle this issue. The top k similar neighbors of each vertex are connected with itself. A similar matrix is calculated by exploiting different methods, and thus, the graph characteristics can be preserved as much as possible.

Recently, researchers have designed more sophisticated models. The performance of earlier matrix factorization model locality preserving projection (LPP) can be improved by introducing an anchor taking advantage of anchorgraph-based locality preserving projection (AgLPP) [74], [75]. The graph structure can be captured by using a local regression model and a global regression process based on local and global regressive mapping (LGRM) [76]. The global geometry can be preserved by using local spline regression [77].

More information can be preserved by exploiting the auxiliary information. An adjacency graph and a labeled graph were constructed in [78]. The objective function of LPP preserves the local geometric structure of the datasets [67]. An adjacency graph and a relational feedback graph were constructed in [79] as well. The graph Laplacian regularization, k-means, and PCA were considered in RF-semi-NMF-PCA simultaneously [80]. Other works, e.g., [81], adopt semidefinite programming to learn the adjacency graph that maximizes the pairwise distances.

2) Vertex Proximity Matrix Factorization: Apart from solving the above generalized eigenvalue problem, another approach of matrix factorization is to factorize vertex proximity matrix directly. In general, matrix factorization can be used to learn the graph structure from nonrelational data, and it is applicable to learn homogeneous graphs.

Based on matrix factorization, vertex proximity can be approximated in a low dimensional space. The objective of preserving vertex proximity is to minimize the error. The singular value decomposition (SVD) of vertex proximity matrix was adopted in [82]. There are some other approaches such as regularized Gaussian matrix factorization [83], low-rank matrix factorization [84], for solving SVD.

3) Discussion: Matrix factorization algorithms operate on an interaction matrix to decompose several lower dimension matrices. The process brings some drawbacks. For example, the algorithms require a large memory when the decomposed matrices become large. In addition, matrix factorization algorithms

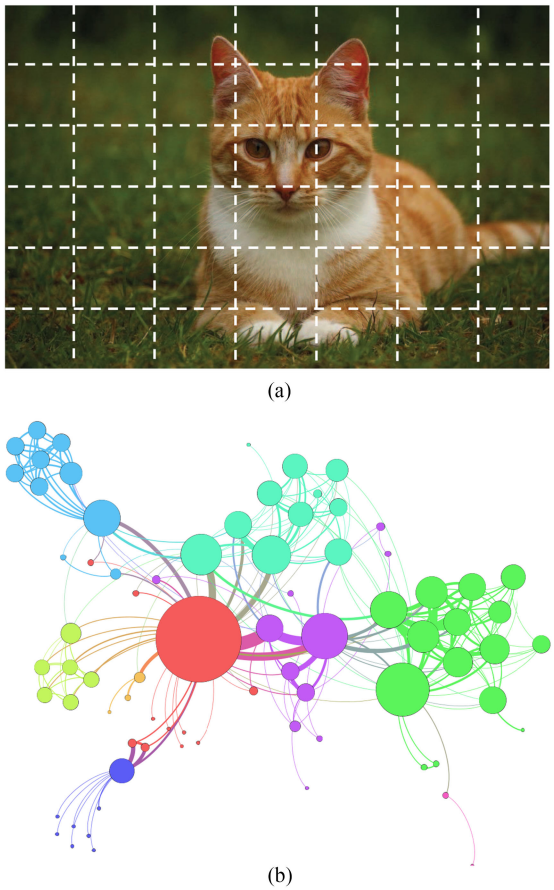


Fig. 5. Example of NRL mapping an image from Euclidean space into non-Euclidean space. (a) Image in Euclidean space and (b) Graph in non-Euclidean space.

are not applicable to supervised or semisupervised tasks with the training process.

C. Random Walk-Based Methods

Random walk is a convenient and effective way to sample networks [85], [86]. This method can generate sequences of nodes meanwhile preserving original relations between nodes. Based on network structure, NRL can generate feature vectors of vertices so that downstream tasks can mine network information in a low dimensional space. An example of NRL is shown in Fig. 5. The image in Euclidean space is shown in Fig. 5(a), and the corresponding graph in non-Euclidean space is shown in Fig. 5(b). As one of the most successful NRL algorithms, random walks play an important role in dimensionality reduction.

1) Structure-Based Random Walks: Graph-structured data have various data types and structures. The information encoded in a graph is related to graph structure and vertex attributes, which are the two key factors affecting the reasoning of networks. In real-world applications, many networks only have structural information, but lack vertex attribute information. How to identify network structure information effectively, such as important vertices and invisible links, attracts the interest of network scientists [87]. Graph data have high dimensional characteristics. Traditional network analysis methods cannot be used for analyzing graph data in a continuous space.

In recent years, various NRL methods have been proposed, which preserve rich structural information of networks. DeepWalk [88] and Node2vec [7] are two representative methods for generating network representation of basic network topology information. These methods use RFMs to generate random sequences on networks. By treating the vertices as words and the generated random sequences of vertices as word sequences (sentences), the models can learn the embedding representation of the vertices by inputting these sequences into the Word2vec model [89]–[91]. The principle of the learning model is to maximize the co-occurrence probability of vertices such as Word2vec. In addition, Node2vec shows that network has complex structural characteristics, and different network structure samplings can obtain different results. The sampling mode of DeepWalk is not enough to capture the diversity of connection patterns in networks. Node2vec designs a random walk sampling strategy, which can sample the networks with the preference of breadth-first sampling and depth-first sampling by adjusting the parameters.

The NRL algorithms mentioned above focused on the first-order proximity information of vertices. Tang *et al.* [92] proposed a method called LINE for large-scale network embedding. LINE can maintain the first and second order approximations. The first-order neighbor refers to the one-hop neighbor between two vertices, and the second-order neighbor is the neighbor with two hops. LINE is not a deep learning based model, but it is often compared with these edge modeling based methods.

It has been proved that the network structure information plays an important role in various network analysis tasks. In addition to this structural information, network attributes in the original network space are also critical in modeling the formation and evolution of the network [93].

2) Structure and Vertex Information-Based Random Walks: In addition to network topology, many types of networks also have rich vertex information, such as vertex content or label in networks. Yang *et al.* [84] proposed an algorithm called TADW. The model is based on DeepWalk and considers the text information of vertices. The MMDW [94] is another model based on DeepWalk, which is a kind of semisupervised network embedding algorithm, by leveraging labeling information of vertices to enhance the performance. Focusing on the structural identity of nodes, Ribeiro *et al.* [95] formulated a framework named Struc2vec. The framework considers nodes with similar local structure rather than neighborhood and labels of nodes. With hierarchy to evaluate structural similarity, the framework constrains structural similarity more stringently. The experiments indicate that DeepWalk and Node2vec are worse than Struc2vec which considers structural identity. There are some other NRL models, such as Planetoid [96], which learn network representation using the feature of network structure and vertex attribute information. It is well known that vertex attributes provide effective information for improving network representation and help to learn embedded vector space. In the case of relatively sparse network topology, vertex attribute information can be used as supplementary information to improve the accuracy

of representation. In practice, how to use vertex information effectively and how to apply this information to network vertex embedding are the main challenges in NRL.

Researchers not only investigate random walk based NRL on vertices but also on graphs. Adhikari *et al.* [97] proposed an unsupervised scalable algorithm, Sub2Vec, to learn arbitrary subgraph. To be more specific, they proposed a method to measure the similarities between subgraphs without disturbing local proximity. Narayanan *et al.* [98] proposed graph2vec, which is a neural embedding framework. Modeling on neural document embedding models, graph2vec takes a graph as a document and the subgraph around words as vertices. By migrating the model to graphs, the performance of graph2vec significantly exceeds other substructure representation learning algorithms.

Generally, random walk can be regarded as a Markov process. The next state of the process is only related to last state, which is known as Markov chain. Inspired by vertex-reinforced random walks, Benson *et al.* [99] presented spacey random walk, a non-Markovian stochastic process. As a specific type of a more general class of vertex-reinforced random walks, it takes the view that the probability of time remained on each vertex relates to the long term behavior of dynamical systems. They proved that dynamical systems can converge to a stationary distribution under sufficient conditions.

Recently, with the development of generative adversarial network (GAN), researchers combined random walks with the GAN method [100], [101]. Existing research on NRL can be divided into generative models and discriminative models. GraphGAN [100] integrated these two kinds of models and played a game-theoretical minimax game. With the process of the game, the performance of the two models can be strengthened. Random walk is used as a generator in the game. NetGAN [101] is a generative model that can model network in real applications. The method takes the distribution of biased random walk as input, and can produce graphs with known patterns. It preserves important topology properties and does not need to define them in model definition.

3) Random Walks in Heterogeneous Networks: In reality, most networks contain more than one type of vertex, and hence networks are heterogeneous. Different from homogeneous NRL, heterogenous NRL should well reserve various relationships among different vertices [102]. Considering the ubiquitous existence of heterogeneous networks, many efforts have been made to learn network representations of heterogeneous networks. Compared to homogeneous NRL, the proximity among entities in heterogeneous NRL is more than a simple measure of distance or closeness. The semantics among vertices and links should be considered. Some typical scenarios include knowledge graphs and social networks.

Knowledge graph is a popular research domain in recent years. A vital part in knowledge base population is relational inference. The central problem of relational inference is inferring unknown knowledge from the existing facts in knowledge bases [103]. There are three types of common relational inference method in general: statistical relational learning (SRL), latent factor models (LFMs), and random walk models (RWMs). Relational learning methods based on statistics lack generality and scalability. As a result, LFM-based graph embedding and relational paths-based random walk have been adopted more widely.

In a knowledge graph, there exist various vertices and various types of relationships among different vertices. For example, in a scholar related knowledge graph [2], [28], the types of vertices include scholar, paper, publication venue, institution, etc. The types of relationships include coauthor, citation, publication, etc. The key idea of knowledge graph embedding is to embed vertices and their relationships into a low dimensional vector space, while the inherent structure of the knowledge graph can be reserved [104].

For relational paths-based random walk, the path ranking algorithm (PRA) is a path finding method using random walks to generate relational features on graph data [105]. Random walks in PRA are with restart, and combine features with a logistic regression. However, PRA cannot predict connection between two vertices if there does not exist a path between them. Gardner *et al.* [106], [107] introduced two ways to improve the performance of PRA. One method enables more efficient processing to incorporate new corpus into knowledge base, while the other method uses vector space to reduce the sparsity of surface forms. To resolve cascade errors in knowledge construction, Wang and Cohen [108] proposed a joint information extraction and knowledge base based model with a recursive random walk. Using latent context of the text, the model obtains additional improvement. Liu *et al.* [109] developed a new random walk based learning algorithm named hierarchical random-walk inference (HiRi). It is a two-tier scheme: the upper tier recognizes relational sequence pattern, and the lower tier captures information from subgraphs of knowledge bases.

Another widely investigated type of heterogeneous networks is social networks, such as online social networks and location-based social networks. Social networks are heterogeneous in nature because of the different types of vertices and relations. There are two main ways to embed heterogeneous social networks, including meta path-based approaches and random walk-based approaches.

A meta path in heterogeneous networks is defined as a sequence of vertex types encoding significant composite relations among various types of vertices. Aiming to employ the rich information in social networks by exploiting various types of relationships among vertices, Fu *et al.* [110] proposed HIN2Vec, which is a representation learning framework based on meta paths. HIN2Vec is a neural network model and the meta paths are well embedded based on two independent phases, i.e., training data preparation and representation learning. Experimental results on various social network datasets show that HIN2Vec model is able to automatically learn vertex vector in heterogeneous networks to support a variety of applications. Metapath2vec [111] was designed by formalizing meta path-based random walks to construct the neighborhoods of a vertex in heterogeneous networks. It takes the advantage of a heterogeneous skip-gram model to perform vertex embedding.

Meta path-based methods require either prior knowledge for optimal meta path selection or extended computations for path length selection. To overcome these challenges, random walk

based approaches have been proposed. Hussein *et al.* [112] proposed the JUST model, which is a heterogeneous graph embedding approach using random walks with jump and stay strategies so that the aforementioned bias can be overcomed effectively. Another method which does not require prior knowledge for meta path definition is MPDRL [113], meta path discovery with reinforcement earning. This method employs the reinforcement learning algorithm to perform multihop reasoning to generate path instances and then further summarizes the important meta paths using the lowest common ancestor principle. Shi *et al.* [114] proposed the HERec model, which utilizes the heterogeneous information network embedding for providing accurate recommendations in social networks. HERec is designed based on a random walk-based approach for generating meaningful vertex sequences for heterogeneous network embedding. HERec can effectively adopt the auxiliary information in heterogeneous information networks. Other typical heterogeneous social network embedding approaches include, e.g., PTE [115] and SHNE [116].

4) Random Walks in Time-Varying Networks: Network is evolving over time, which means that new vertices may emerge and new relations may appear. Therefore, it is significant to capture the temporal behaviour of networks in network analysis. Many efforts have been made to learn time-varying network embedding (e.g., dynamic networks or temporal networks) [117]. In contrast to static network embedding, time-varying NRL should consider the network dynamics, which means that old relationships may become invalid and new links may appear.

The key of time-varying NRL is to find a suitable way to incorporate the time characteristic into embedding via reasonable updating approaches. Nguyen *et al.* [118] proposed the CTDNE model for continuous dynamic network embedding based on random walk with "chronological" paths which can only move forward as time goes on. Their model is more suitable for time-dependent network representation that can capture the important temporal characteristics of continuous-time dynamic networks. Results on various datasets show that CTDNE outperforms static NRL approaches. Zuo *et al.* [119] proposed the HTNE model which is a temporal NRL approach based on the Hawkes process. HTNE can well integrate the Hawkes process into network embedding so that the influence of historical neighbors on the current neighbors can be accurately captured.

For unseen vertices in a dynamical network, GraphSAGE [120] was presented to efficiently generate embeddings for new vertices in network. In contrast to methods that training embedding for every vertex in the network, GraphSAGE designs a function to generate embedding for a vertex with features of the neighborhoods locally. After sampling neighbors of a vertex, GraphSAGE uses different aggregators to update the embedding of the vertex. However, current graph neural methods are proficient of only learning local neighborhood information and cannot directly explore the higher order proximity and the community structure of graphs.

5) Discussion: As mentioned before, random walk is a fundamental way to sample networks. The sequences of nodes could preserve the information of network structure. However, there are some disadvantages of this method. For example, random walk relies on random strategies, which creates some uncertain relations of nodes. To reduce this uncertainty, it needs to increase the number of samples, which will significantly increase the complexity of algorithms. Some random walk variants could preserve local and global information of networks, but they might not be effective in adjusting parameters to adapt to different types of networks.

D. Deep Learning on Graphs

Deep learning is one of the hottest areas over the past few years. Nevertheless, it is an attractive and challenging task to extend the existing neural network models, such as recurrent neural networks (RNNs) or convolutional neural networks (CNNs), to graph data. Gori *et al.* [121] proposed a GNN model based on recursive neural network. In this model, a transfer function is implemented, which maps the graph or its vertices to an m-dimensional Euclidean space. In recent years, lots of GNN models have been proposed.

1) Graph Convolutional Networks: GCN works on the basis of grid structure domain and graph structure domain [122].

a) Time domain and spectral methods: Convolution is one of a common operation in deep learning. However, since graph lacks a grid structure, standard convolution over images or text cannot be directly applied to graphs. Bruna *et al.* [122] extended the CNN algorithm from image processing to the graph using the graph Laplacian matrix, dubbed as spectral graph CNN. The main idea is similar to Fourier basis for signal processing. Based on [122], Henaff *et al.* [123] defined kernels to reduced the learning parameters by analogizing the local connection of CNNs on the image. Defferrard *et al.* [124] provided two ways for generalizing CNNs to graph structure data based on graph theory. One method is to reduce the parameters by using polynomial kernel, and this method can be accelerated by using Chebyshev polynomial approximation. The other method is the special pooling method, which is pooling on the binary tree constructed from vertices. An improved version of [124] was introduced by Kipf and Welling [125]. The proposed method is a semisupervised learning method for graphs. The algorithm employs an excellent and straightforward neural network followed by a layer-by-layer propagation rule, which is based on the first-order approximation of spectral convolution on the graph and can be directly acted on the graph.

There are some other time domain based methods. Based on the mixture model of CNNs, for instance, Monti *et al.* [126] generalized the CNN to non-Euclidean space. Zhou and Li [127] proposed a new CNN graph modeling framework, which designs two modules for graph structure data: K-order convolution operator and adaptive filtering module. In addition, the high-order adaptive graph convolution network (HA-GCN) framework proposed in [127] is a general architecture that is suitable for many applications of vertices and graph centers. Manessi *et al.* [128] proposed a dynamic graph convolution network algorithm for dynamic graphs. The core idea of the algorithm is to combine the expansion of graph convolution with the improved long short term-memory networks (LSTM) algorithm, and then train and learn the downstream recursive unit by using graph structure

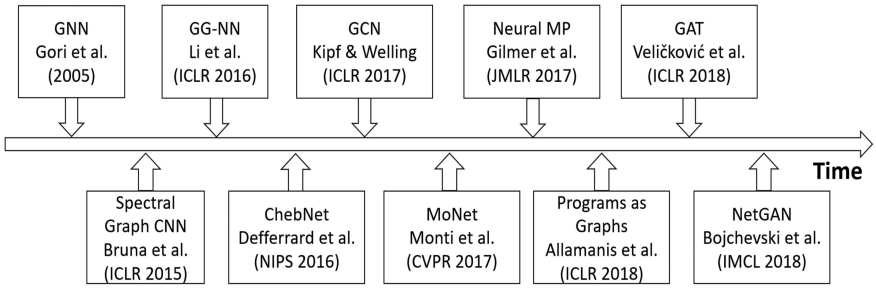


Fig. 6. Brief history of algorithms of deep learning on graphs.

data and vertex features. The spectral-based NRL methods have many applications, such as vertex classification [125], traffic forecasting [129], [130], and action recognition [131].

b) Space domain and spatial methods: Spectral graph theory provides a convolution method on graphs, but many NRL methods directly use convolution operation on graphs in space domain. Niepert *et al.* [132] applied graph labeling procedures such as Weisfeiler–Lehman kernel on graphs to generate unique order of vertices. The generated subgraphs can be fed to the traditional CNN operation in space domain. Duvenaud *et al.* [133] designed neural fingerprints (FP), which is a spatial method using the first-order neighbors similar to the GCN algorithm. Atwood and Towsley [134] proposed another convolution method, called diffusion-CNN, which incorporates transfer probability matrix and replaces the characteristic basis of convolution with diffusion basis. Gilmer *et al.* [135] reformulated existing models into a single common framework, and exploited this framework to discover new variations. Allamanis *et al.* [136] represented the structure of code from syntactic and semantic, and utilized the GNN method to recognize program structures.

Zhuang and Ma [137] designed dual graph convolution networks (DGCNs), which use diffusion basis and adjacency basis. DGCN uses two convolutions: one is the characteristic form of polynomial filter, and the other is to replace the adjacency matrix with the positive pointwise mutual information (PPMI) of the transition probability [89]. Dai *et al.* [138] proposed the SSE algorithm, which uses asynchronous random to learn vertex representation so as to improve learning efficiency. In this model, a recursive method is adopted to learn vertex latent representation and the sampled batch data are utilized to update parameters. The recursive function of SSE is calculated from the weighted average of historical state and new state. Zhu *et al.* [139] proposed a graph smoothing splines neural network which exploits nonsmoothing node features and global topological knowledge such as centrality for graph classification. Gao *et al.* [140] proposed a large scale graph convolution network (LGCN) based on vertex feature information. In order to adapt to the scene of large scale graphs, they proposed a subgraph training strategy, which first trained the sampled subgraph in a small batch. Based on a deep generative graph model, a novel method called DeepNC for inferring the missing parts of a network was proposed in [141].

A brief history of deep learning on graphs is shown in Fig. 6. GNN has attracted lots of attention since 2015, and it is widely studied and used in various fields.

2) Graph Attention Networks: In sequence-based tasks, attention mechanism has been regarded as a standard [142]. GNNs achieve lots of benefits from the expanded model capacity of attention mechanisms. GATs are a kind of spatial-based GCNs [143]. It takes the attention mechanism into consideration when determining the weights of vertex's neighbors. Likewise, gated attention networks (GAANs) also introduced the multi-head attention mechanism for updating the hidden state of some vertices [144]. Unlike GATs, GAANs employ a self-attention mechanism which can compute different weights for different heads. Some other models such as graph attention model (GAM) were proposed for solving different problems [145]. Take GAM as an example, the purpose of GAM is to handle graph classification. Therefore, GAM is set to process informative parts by visiting a sequence of significant vertices adaptively. The model of GAM contains LSTM network, and some parameters contain historical information, policies, and other information generated from exploration of the graph. Attention walks (AWs) are another kind of learning model based on GNN and random walks [146]. In contrast to DeepWalk, AWs use differentiable attention weights when factorizing the co-occurrence matrix [88].

3) Graph Auto-Encoders: GAE uses GNN structure to embed network vertices into low dimensional vectors. One of the most general solutions is to employ a multilayer perception as the encoder for inputs [147]. Therein the decoder reconstructs neighborhood statistics of the vertex. PPMI or the first and the second nearest neighborhood can be taken into statistics [148], [149]. Deep neural networks for graph representations (DNGR) employ PPMI. Structural deep network embedding (SDNE) employs stacked auto-encoder to maintain both the first-order and the second-order proximity. Auto-encoder [150] is a traditional deep learning model, which can be classified as a self-supervised model [151]. Deep recursive network embedding (DRNE) reconstructs some vertices' hidden state rather than the entire graph [152]. It has been found that if we regard GCN as an encoder, and combine GCN with GAN or LSTM with GAN, then we can design the auto-encoder for graphs. Generally speaking, DNGR and SDNE embed vertices by the given structure features, while other methods such as DRNE learn both topology structure and content features [148], [149]. Variational graph auto-encoder [153] is another successful approach that employs GCN as an encoder and a link prediction layer as a decoder. Its successor, adversarially regularized variational graph auto-encoder [154], adds a regularization process with an adversarial training approach to learn a more robust embedding.

4) Graph Generative Networks: The purpose of graph generative networks is to generate graphs according to the given observed set of graphs. Many previous methods of graph generative networks have their own application domains. For example, in natural language processing, the semantic graph or the knowledge graph is generated based on the given sentences. Some general methods have been proposed recently. One kind of them considers the generation process as the formation of vertices and edges. Another kind is to employ generative adversarial training. Some GCNs-based graph generative networks such as molecular generative adversarial networks (MolGAN) integrate GNN with reinforcement learning [155]. Deep generative models of graphs (DGMG) achieves a hidden representation of existing graphs by utilizing spatial-based GCNs [156]. There are some knowledge graph embedding algorithms based on GAN and zero-shot learning [157]. Vyas *et al.* [158] proposed a generalized zero-shot learning model, which can find unseen semantic in knowledge graphs.

5) Graph Spatial-Temporal Networks: Graph spatial-temporal networks simultaneously capture the spatial and temporal dependence of graphs. The global structure is included in the spatial-temporal graphs, and the input of each vertex varies with the change of time. For example, in traffic networks, each sensor records the traffic speed of a road continuously as a vertex, in which the edge of the traffic networks is determined by the distance between the sensor pairs [129]. The goal of a spatial-temporal network can be to predict future vertex values or labels, or to predict spatial-temporal graph labels. Recent studies in this direction have discussed the use of GCNs, the combination of GCNs with RNN or CNN, and recursive structures for graph structures [130], [131], [159].

6) Discussion: In this context, the task of graph learning can be seen as optimizing the objective function by using gradient descent algorithms. Therefore the performance of deep learning based NRL models is influenced by gradient descent algorithms. They may encounter challenges like local optimal solutions and the vanishing gradient problem.

III. Applications

Many problems can be solved by graph learning methods, including supervised, semisupervised, unsupervised, and reinforcement learning. Some researchers classify the applications of graph learning into three categories, i.e., structural scenarios, nonstructural scenarios, and other application scenarios [18]. Structural scenarios refer to the situation where data are performed in explicit relational structures, such as physical systems, molecular structures, and knowledge graphs. Nonstructural scenarios refer to the situation where data are with unclear relational structures, such as images and texts. Other application scenarios include, e.g., integrating models and combinatorial optimization problems. Table II lists the neural components and applications of various graph learning methods.

A. Datasets and Open-Source Libraries

There are several datasets and benchmarks used to evaluate the performance of graph learning approaches for various tasks such as link prediction, node classification, and graph visualization. For instance, datasets like Cora[1](citation network), Pubmed[2](citation network), BlogCatalog[3](social network), Wikipedia[4] (language network), and PPI[5] (biological network) include nodes, edges, labels, or attributes of nodes. Some research institutions developed graph learning libraries, which include common and classical graph learning algorithms. For example, OpenKE[6] is a Python library for knowledge graph embedding based on PyTorch. The open-source framework has the implementations of RESCAL, HolE, DistMult, ComplEx, etc. CogDL[7] is a graph representation learning framework, which can be used for node classification, link prediction, graph classification, etc.

B. Text

Many data are in textual form coming from various resources like web pages, emails, documents (technical and corporate), books, digital libraries and customer complains, letters, patents, etc. Textual data are not well structured for obtaining any meaningful information as text often contains rich context information. There exist abundant applications around text, including text classification, sequence labeling, sentiment classification, etc. Text classification is one of the most classical problems in natural language processing. Popular algorithms proposed to handle this problem include GCNs [120], [125], GATs [143], Text GCNs [160], and sentence LSTM [161]. Sentence LSTM has also been applied to sequence labeling, text generation, multihop reading comprehension, etc [161]. Syntactic GCN was proposed to solve semantic role labeling and neural machine translation [162]. Gated graph neural networks (GGNNs) can also be used to address neural machine translation and text generation [163]. For relational extraction, Tree LSTM, graph LSTM, and GCN are better solutions [164].

C. Images

Graph learning applications pertaining to images include social relationship understanding, image classification, visual question answering (VQA), object detection, region classification, and semantic segmentation, etc. For social relationship understanding, for instance, graph reasoning model (GRM) is widely used [165]. Since social relationships such as friendships are the basis of social networks in real world, automatically interpreting these relationships is important for understanding human behaviors. GRM introduces GGNNs to learn a propagation mechanism. Image classification is a classical problem, in which GNNs have demonstrated promising performance. VQA is a learning task that involves both computer vision and

[1][Online]. Available: https://relational.fit.cvut.cz/dataset/CORA
[2][Online]. Available: https://catalog.data.gov/dataset/pubmed
[3][Online]. Available: http://networkrepository.com/soc-BlogCatalog.php
[4][Online]. Available: https://en.wikipedia.org/wiki/Wikipedia:Database_download
[5][Online]. Available: https://openwetware.org/wiki/Protein-protein_interaction_databases
[6][Online]. Available: https://github.com/thunlp/OpenKE
[7][Online]. Available: https://github.com/THUDM/cogdl/

TABLE II
SUMMARY OF GRAPH LEARNING METHODS AND THEIR APPLICATIONS

Categories	Algorithms	Neural Component	Applications
Time Domain and Spectral Methods	SNLCN [122]	Graph Neural Network	Classification
	DCN [123]	Spectral Network	Classification
	ChebNet [124]	Convolution Network	Classification
	GCN [125]	Spectral Network	Classification
	HA-GCN [127]	GCN	Classification
	Dynamic GCN [128]	GCN, LSTM	Classification
	DCRNN [129]	Diffusion Convolution Network	Traffic Forecasting
	ST-GCN [131]	GCN	Action Recognition
Space Domain and Spatial Methods	PATCHY-SAN [132]	Convolutional Network	Runtime Analysis, Feature Visualization, Graph Classification
	Neural FP [133]		Sub-graph Classification
	DCNN [134]	DCNN	Classification
	DGCN [137]	Graph-Structure-Based Convolution, PPMI-Based Convolution.	Classification
	SSE [138]		Vertex Classification
	LGCN [140]	Convolutional Neural Network	Vertex Classification
	STGCN [130]	Gated Sequential Convolution	Traffic Forecasting
Deep Learning Model Based Methods	GATs [143]	Attention Neural Network	Classification
	GAAN [144]		Vertex Classification
	GAM [145]		Graph Classification
	Aws [146]	Auto-encoder Neural Network	Link Prediction, Sensitivity Analysis, Vertex Classification
	SDNE [149]		Classification, Link Prediction, Visualization
	DNGR [148]		Clustering, Visualization
	DRNE [152]		Regular Equivalence Prediction, Structural Role Classification, Network Visualization
	MolGAN [155]	Generative Neural Network	Generative Model
	DGMG [156]		Molecule Generation
	DCRNN [129]	Diffusion Convolution Network	Traffic Forecasting
	STGCN [130]	Gated Sequential Convolution	
	ST-GCN [131]	GCNs	Action Recognition

natural language processing. A VQA system takes the form of a certain pictures and its open natural language question as input, in order to generate a natural language answer as output. Generally speaking, VQA is question-and-answer for a given picture. GGNNs have been exploited to help with VQA [166].

D. Science

Graph learning has been widely adopted in science. Modeling real-world physical systems is one of the most fundamental perspectives in understanding human intelligence. Representing objects as vertices and relations as edges between them is a simple but effective way to perform physics. Battaglia *et al.* [167] proposed interaction networks (INs) to predict and infer abundant physical systems, in which IN takes objects and relationships as input. Based on IN, the interactions can be reasoned and the effects can be applied. Therefore, physical dynamics can be predicted. Visual interaction networks (VINs) can make predictions from pixels by first learning a state code from two continuous input frames per object [168].

Other graph networks-based models have been developed to address chemistry and biology problems. Calculating molecular fingerprints, i.e., using feature vectors to represent molecular,

is a central step. Researchers [169] proposed neural graph fingerprints using GCNs to calculate substructure feature vectors. Some studies focused on protein interface prediction. This is a challenging issue with significant applications in biology. Besides, GNNs can be used in biomedical engineering as well. Based on protein–protein INs, Rhee *et al.* [170] used graph convolution and protein relation networks to classify breast cancer subtypes.

E. Knowledge Graphs

Various heterogeneous objects and relationships are regarded as the basis for a knowledge graph [171]. GNNs can be applied in knowledge base completion (KBC) for solving the out-of-knowledge-base (OOKB) entity problem [172]. The OOKB entities are connected to existing entities. Therefore, the embedding of OOKB entities can be aggregated from existing entities. Such kind of algorithms achieve reasonable performance in both settings of KBC and OOKB. Likewise, GCNs can also be used to solve the problem of cross-lingual knowledge graph alignment. The main idea of the model is to embed entities from different languages into an integrated embedding space. Then the model aligns these entities according to their embedding similarities.

Generally speaking, knowledge graph embedding can be categorized into two types: translational distance models and semantic matching models. Translational distance models aim to learn the low dimensional vector of entities in a knowledge graph by employing distance-based scoring functions. These methods calculate the plausibility as the distance between two entities after a translation measured by the relationships between them. Among current translational distance models, TransE [173] is the most influential one. TransE can model the relationship of entities by interpreting them as translations operating on the low dimensional embedding. Inspired by TranE, TranH [174] was proposed to overcome the disadvantages of TransE in dealing with 1-to-N, N-to-1, and N-to-N relations by introducing relation-specific hyperplanes. Instead of hyperplanes, TransR [175] introduces relation-specific spaces to solve the flows in TransE. Meanwhile, various extensions of TransE have been proposed to enhance knowledge graph embeddings, such as TransD [176] and TransF [177]. On the basis of TransE, DeepPath [178] incorporates reinforcement learning methods for learning relational paths in knowledge graphs. By designing a complex reward function involving accuracy, efficiency and path diversity, the path finding process is better controlled and more flexible.

Semantic matching models utilize the similarity-based scoring functions. They measure the plausibility among entities by matching latent semantics of entities and relations in low dimensional vector space. Typical models of this type include RESCAL [179], DistMult [180], ANALOGY [181], etc.

F. Combinatorial Optimization

Classical problems such as traveling salesman problem (TSP) and minimum spanning tree (MST) have been solved by using different heuristic solutions. Recently, deep neural networks have been applied to these problems. Some solutions make further use of GNNs thanks to their structures. Bello *et al.* [182] first proposed such kind of methods to solve TSP. Their method mainly contains two steps, i.e., a parameterized reward pointer network and a strategy gradient module for training. Khalil *et al.* [183] improved this work with GNN and achieved better performance by two main procedures. First, they used structure2vec to achieve vertex embedding and then input them into Q-learning module for decision-making. This work also proves the embedding ability of GNN. Nowak *et al.* [184] focused on the secondary assignment problem, i.e., measuring the similarity of two graphs. The GNN model learns each graph's vertex embedding and uses the attention mechanism to match the two graphs. Other studies use GNNs directly as the classifiers, which can perform the intensive prediction on graphs with two sides. The rest of the model facilitates diverse choices and effective training.

IV. Open Issues

In this section, we briefly summarize several future research directions and open issues for graph learning.

A. Dynamic Graph Learning

For the purpose of graph learning, most existing algorithms are suitable for static networks without specific constraints. However, dynamic networks such as traffic networks vary over time. Therefore, they are hard to deal with. Dynamic graph learning algorithms have rarely been studied in the literature. It is of significant importance that dynamic graph learning algorithms are designed to maintain good performance, especially in the case of dynamic graphs.

B. Generative Graph Learning

Inspired by the GANs, generative graph learning algorithms can unify the generative and discriminative models by playing a game-theoretical min–max game. This generative graph learning method can be used for link prediction, network evolution, and recommendation by boosting the performance of generative and discriminative models alternately and iteratively.

C. Fair Graph Learning

Most graph learning algorithms rely on deep neural networks, and the resulting vectors may have captured undesired sensitive information. The bias existing in the network is reinforced, and hence it is of significant importance to integrate the fair metrics into the graph learning algorithms to address the inherent bias issue.

D. Interpretable Graph Learning

The models of graph learning are generally complex by incorporating both graph structure and feature information. The interpretability of graph learning (based) algorithms remains unsolved since the structures of graph learning algorithms are still a black box. For example, drug discovery can be achieved by graph learning algorithms. However, it is unknown how this drug is discovered as well as the reason behind this discovery.

The interpretability behind graph learning needs to be further studied.

V. Conclusion

This article gives a general description of graph learning, and provides a comprehensive review of the state-of-the-art graph learning methods. We examine existing graph learning methods under four categories: GSP-based methods, matrix factorization-based methods, random walk-based methods, and deep learning-based methods. The presented applications of graph learning methods are mainly under these four categories in areas such as text, images, science, knowledge graphs, and combinatorial optimization. We also discuss some future research directions in the field of graph learning. Graph learning is currently growing at an unprecedented speed. We do hope that this survey will help researchers and practitioners with their research and development in graph learning and related areas.

Acknowledgment

The authors would like to thank Prof. H. Abbass at the University of New South Wales, Y. Sun, J. Liu, H. Ren at the Dalian University of Technology, and anonymous reviewers for their valuable comments and suggestions.

References

[1] S. Fortunato *et al.*, "Science of science," *Science*, vol. 359, no. 6379, 2018, Art no. eaao0185.
[2] J. Liu, J. Ren, W. Zheng, L. Chi, I. Lee, and F. Xia, "Web of scholars: A scholar knowledge graph," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, Jul. 2020, pp. 2153–2156.
[3] J. Liu *et al.*, "Shifu2: A network representation learning based model for advisor-advisee relationship mining," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 4, pp. 1763–1777, Apr. 2021.
[4] D. Zhang, J. Yin, X. Zhu, and C. Zhang, "Network representation learning: A survey," *IEEE Trans. Big Data*, vol. 6, no. 1, pp. 3–28, Mar. 2020.
[5] K. Sun, J. Liu, S. Yu, B. Xu, and F. Xia, "Graph force learning," in *Proc. IEEE Int. Conf. Big Data*, 2020, pp. 2987–2994.
[6] F. Xia, J. Wang, X. Kong, D. Zhang, and Z. Wang, "Ranking station importance with human mobility patterns using subway network datasets," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 7, pp. 2840–2852, Jul. 2020.
[7] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proc. 22nd ACM Int. Conf. Knowl. Discov. Data Mining*, Aug. 2016, pp. 855–864.
[8] K. Sun, L. Wang, B. Xu, W. Zhao, S. W. Teng, and F. Xia, "Network representation learning: From traditional feature learning to deep learning," *IEEE Access*, vol. 8, no. 1, pp. 205600–205617 , Nov. 2020.
[9] S. Yu, F. Xia, J. Xu, Z. Chen, and I. Lee, "Offer: A Motif dimensional framework for network representation learning," in *Proc. 29th ACM Int. Conf. Inf. Knowl. Manage.*, 2020, pp. 3349–3352.
[10] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013.
[11] T. Guo *et al.*, "Graduate employment prediction with bias," in *Proc. 34th AAAI Conf. Artif. Intell.*, 2020, pp. 670–677.
[12] F. Xia, A. M. Ahmed, L. T. Yang, J. Ma, and J. J. Rodrigues, "Exploiting social relationship to enable efficient replica allocation in ad-hoc social networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 12, pp. 3167–3176, Dec. 2014.
[13] J. Zhang, W. Wang, F. Xia, Y.-R. Lin, and H. Tong, "Data-driven computational social science: A survey," *Big Data Res.*, vol. 21, Aug. 2020, Art. no. 100145.
[14] F. Xia, A. M. Ahmed, L. T. Yang, and Z. Luo, "Community-based event dissemination with optimal load balancing," *IEEE Trans. Comput.*, vol. 64, no. 7, pp. 1857–1869, Jul. 2015.
[15] F. Xia, H. Wei, S. Yu, D. Zhang, and B. Xu, "A survey of measures for network motifs," *IEEE Access*, vol. 7, no. 1, pp. 106576–106587, 2019.
[16] J. Leskovec and C. Faloutsos, "Sampling from large graphs," in *Proc. 12th ACM Int. Conf. Knowl. Discov. Data Mining*, Aug. 2006, pp. 631–636.
[17] X. Wang, P. Liu, and Y. Gu, "Local-set-based graph signal reconstruction," *IEEE Trans. Signal Process.*, vol. 63, no. 9, pp. 2432–2444, Mar. 2015.
[18] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," *IEEE Trans. Knowl. Data Eng.*, doi: 10.1109/TKDE.2020.2981333, Mar. 2020.
[19] J. Xu *et al.*, "Multivariate relations aggregation learning in social networks," in *Proc. ACM/IEEE Joint Conf. Digit. Libraries*, Aug. 2020, pp. 77–86.
[20] H. D. Bedru *et al.*, "Big networks: A survey," *Comput. Sci. Rev.*, vol. 37, Aug. 2020, Art. no. 100247.
[21] P. Goyal and E. Ferrara, "Graph embedding techniques, applications, and performance: A survey," *Knowl.-Based Syst.*, vol. 151, pp. 78–94, Jul. 2018.
[22] H. Cai, V. W. Zheng, and K. C.-C. Chang, "A comprehensive survey of graph embedding: Problems, techniques, and applications," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 9, pp. 1616–1637, Feb. 2018.
[23] M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich, "A review of relational machine learning for knowledge graphs," *Proc. IEEE Proc. IRE*, vol. 104, no. 1, pp. 11–33, Jan. 2016.
[24] L. Akoglu, H. Tong, and D. Koutra, "Graph based anomaly detection and description: A survey," *Data Mining Knowl. Discov.*, vol. 29, no. 3, pp. 626–688, May 2015.
[25] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2021.
[26] A. Ortega, P. Frossard, J. Kovačević, J. M. Moura, and P. Vandergheynst, "Graph signal processing: Overview, challenges, and applications," *Proc. IEEE Proc. IRE*, vol. 106, no. 5, pp. 808–828, May 2018.
[27] W. Huang *et al.*, "A graph signal processing perspective on functional brain imaging," *Proc. IEEE Proc. IRE*, vol. 106, no. 5, pp. 868–885, May 2018.
[28] F. Xia, W. Wang, T. M. Bekele, and H. Liu, "Big scholarly data: A survey," *IEEE Trans. Big Data*, vol. 3, no. 1, pp. 18–35, Mar. 2017.
[29] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553,2015, Art. no. 436.
[30] J. Liu *et al.*, "Artificial intelligence in the 21st century," *IEEE Access*, vol. 6, pp. 34403–34421, 2018.
[31] A. Sandryhaila and J. M. Moura, "Discrete signal processing on graphs," *IEEE Trans. Signal Process.*, vol. 61, no. 7, pp. 1644–1656, Apr. 2013.
[32] D. Shuman, S. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Process. Mag.*, vol. 3, no. 30, pp. 83–98, May 2013.
[33] M. Puschel and J. M. Moura, "Algebraic signal processing theory: Foundation and 1-D time," *IEEE Trans. Signal Process.*, vol. 56, no. 8, pp. 3572–3585, Aug. 2008.
[34] A. Sandryhaila and J. M. Moura, "Discrete signal processing on graphs: Graph filters," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2013, pp. 6163–6166.
[35] S. Chen, R. Varma, A. Sandryhaila, and J. Kovačević, "Discrete signal processing on graphs: Sampling theory," *IEEE Trans. Signal Process.*, vol. 63, no. 24, pp. 6510–6523, Dec. 2015.
[36] U. V. Luxburg, "A tutorial on spectral clustering," *Statist. Comput.*, vol. 17, no. 4, pp. 395–416, Dec. 2007.
[37] X. Zhu and M. Rabbat, "Graph spectral compressed sensing for sensor networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2012, pp. 2865–2868.
[38] A. Gavili and X.-P. Zhang, "On the shift operator, graph frequency, and optimal filtering in graph signal processing," *IEEE Trans. Signal Process.*, vol. 65, no. 23, pp. 6303–6318, Dec. 2017.
[39] B. Pasdeloup, M. Rabbat, V. Gripon, D. Pastor, and G. Mercier, "Graph reconstruction from the observation of diffused signals," in *Proc. 53rd Annu. Allerton Conf. Commun., Control, Comput.*, 2015, pp. 1386–1390.
[40] A. Anis, A. Gadde, and A. Ortega, "Towards a sampling theorem for signals on arbitrary graphs," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2014, pp. 3864–3868.
[41] G. Puy, N. Tremblay, R. Gribonval, and P. Vandergheynst, "Random sampling of bandlimited signals on graphs," *Appl. Comput. Harmon. Anal.*, vol. 44, no. 2, pp. 446–475, Mar. 2018.

[42] H. Shomorony and A. S. Avestimehr, "Sampling large data on graphs," in *Proc. IEEE Glob. Conf. Signal Inf. Process.*, 2014, pp. 933–936.

[43] L. F. Chamon and A. Ribeiro, "Greedy sampling of graph signals," *IEEE Trans. Signal Process.*, vol. 66, no. 1, pp. 34–47, Jan. 2018.

[44] A. G. Marques, S. Segarra, G. Leus, and A. Ribeiro, "Sampling of graph signals with successive local aggregations," *IEEE Trans. Signal Process.*, vol. 64, no. 7, pp. 1832–1843, Apr. 2016.

[45] S. K. Narang, A. Gadde, and A. Ortega, "Signal processing techniques for interpolation in graph structured data," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2013, pp. 5445–5449.

[46] A. Gadde and A. Ortega, "A probabilistic interpretation of sampling theory of graph signals," in *Proc. IEEE Int. Conf. Acoustics, Speech Signal Process.*, 2015, pp. 3257–3261.

[47] X. Wang, M. Wang, and Y. Gu, "A distributed tracking algorithm for reconstruction of graph signals," *IEEE J. Sel. Topics Signal Process.*, vol. 9, no. 4, pp. 728–740, Jun. 2015.

[48] P. Di Lorenzo, S. Barbarossa, P. Banelli, and S. Sardellitti, "Adaptive least mean squares estimation of graph signals," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 2, no. 4, pp. 555–568, Dec. 2016.

[49] D. Romero, M. Ma, and G. B. Giannakis, "Kernel-based reconstruction of graph signals." *IEEE Trans. Signal Process.*, vol. 65, no. 3, pp. 764–778, Feb. 2017.

[50] M. Nagahara, "Discrete signal reconstruction by sum of absolute values," *IEEE Signal Process. Lett.*, vol. 22, no. 10, pp. 1575–1579, Oct. 2015.

[51] S. Chen, R. Varma, A. Singh, and J. Kovačević, "Signal representations on graphs: Tools and applications,"2015, *arXiv:1512.05406*.

[52] S. Segarra, A. G. Marques, G. Leus, and A. Ribeiro, "Reconstruction of graph signals through percolation from seeding nodes," *IEEE Trans. Signal Process.*, vol. 64, no. 16, pp. 4363–4378, Aug. 2016.

[53] F. Xia, J. Liu, J. Ren, W. Wang, and X. Kong, "Turing number: How far are you to a. m. turing award?" *ACM SIGWEB Newslett.*, vol. Autumn, pp. 1–5, Nov. 2020.

[54] H. E. Egilmez, E. Pavez, and A. Ortega, "Graph learning from data under Laplacian and structural constraints," *IEEE J. Sel. Topics Signal Process.*, vol. 11, no. 6, pp. 825–841, Sep. 2017.

[55] X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst, "Learning Laplacian matrix in smooth graph signal representations," *IEEE Trans. Signal Process.*, vol. 64, no. 23, pp. 6160–6173, Aug. 2016.

[56] V. Kalofolias, "How to learn a graph from smooth signals," in *Proc. Artif. Intell. Statist.*, 2016, pp. 920–929.

[57] E. Pavez and A. Ortega, "Generalized Laplacian precision matrix estimation for graph signal processing," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2016, pp. 6350–6354.

[58] E. Pavez, H. E. Egilmez, and A. Ortega, "Learning graphs with monotone topology properties and multiple connected components," *IEEE Trans. Signal Process.*, vol. 66, no. 9, pp. 2399–2413, May 2018.

[59] B. Pasdeloup, V. Gripon, G. Mercier, D. Pastor, and M. G. Rabbat, "Characterization and inference of graph diffusion processes from observations of stationary signals," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 4, no. 3, pp. 481–496, Sep. 2018.

[60] S. Segarra, A. G. Marques, G. Mateos, and A. Ribeiro, "Network topology inference from spectral templates," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 3, no. 3, pp. 467–483, Sep. 2017.

[61] D. Thanou, X. Dong, D. Kressner, and P. Frossard, "Learning heat diffusion graphs," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 3, no. 3, pp. 484–499, Jul. 2017.

[62] J. Mei and J. M. Moura, "Signal processing on graphs: Causal modeling of unstructured data," *IEEE Trans. Signal Process.*, vol. 65, no. 8, pp. 2077–2092, Apr. 2017.

[63] S. Segarra, G. Mateos, A. G. Marques, and A. Ribeiro, "Blind identification of graph filters," *IEEE Trans. Signal Process.*, vol. 65, no. 5, pp. 1146–1159, Mar. 2017.

[64] F. Xia, N. Y. Asabere, A. M. Ahmed, J. Li, and X. Kong, "Mobile multimedia recommendation in smart communities: A survey," *IEEE Access*, vol. 1, no. 1, pp. 606–624, Sep. 2013.

[65] W. Huang, A. G. Marques, and A. R. Ribeiro, "Rating prediction via graph signal processing," *IEEE Trans. Signal Process.*, vol. 66, no. 19, pp. 5066–5081, Oct. 2018.

[66] F. Xia, H. Liu, I. Lee, and L. Cao, "Scientific article recommendation: Exploiting common author relations and historical preferences," *IEEE Trans. Big Data*, vol. 2, no. 2, pp. 101–112, Jun. 2016.

[67] X. He and P. Niyogi, "Locality preserving projections," in *Proc. Adv. Neural Inf. Process. Syst.*, Dec. 2004, pp. 153–160.

[68] M. Chen, I. W. Tsang, M. Tan, and T. J. Cham, "A unified feature selection framework for graph embedding on high dimensional data," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 6, pp. 1465–1477, Jun. 2015.

[69] S. Yan, D. Xu, B. Zhang, H.-J. Zhang, Q. Yang, and S. Lin, "Graph embedding and extensions: A general framework for dimensionality reduction," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 1, pp. 40–51, Jan. 2007.

[70] I. Borg and P. Groenen, "Modern multidimensional scaling: Theory and applications," *J. Educ. Meas.*, vol. 40, no. 3, pp. 277–280, 2003.

[71] M. Balasubramanian and E. L. Schwartz, "The isomap algorithm and topological stability," *Science*, vol. 295, no. 5552, pp. 7–7, Jan. 2002.

[72] W. N. Anderson Jr and T. D. Morley, "Eigenvalues of the Laplacian of a graph," *Linear Multilinear Algebra*, vol. 18, no. 2, pp. 141–145, Oct. 1985.

[73] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, Dec. 2000.

[74] R. Jiang, W. Fu, L. Wen, S. Hao, and R. Hong, "Dimensionality reduction on anchorgraph with an efficient locality preserving projection," *Neurocomputing*, vol. 187, pp. 109–118, Apr. 2016.

[75] L. Wan, Y. Yuan, F. Xia, and H. Liu, "To your surprise: Identifying serendipitous collaborators," *IEEE Trans. Big Data*, DOI: 10.1109/TBDATA.2019.2921567, Jun. 2019.

[76] Y. Yang, F. Nie, S. Xiang, Y. Zhuang, and W. Wang, "Local and global regressive mapping for manifold learning with out-of-sample extrapolation," in *24th AAAI Conf. Artif. Intell.*, 2010, pp. 649–654.

[77] S. Xiang, F. Nie, C. Zhang, and C. Zhang, "Nonlinear dimensionality reduction with local spline embedding," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1285–1298, Sep. 2009.

[78] D. Cai, X. He, and J. Han, "Spectral regression: A unified subspace learning framework for content-based image retrieval," in *Proc. 15th ACM int. conf. Multimedia*, 2007, pp. 403–412.

[79] X. He, W.-Y. Ma, and H.-J. Zhang, "Learning an image manifold for retrieval," in *Proc. 12th Annu. ACM Int. Conf. Multimedia*, 2004, pp. 17–23.

[80] K. Allab, L. Labiod, and M. Nadif, "A semi-NMF-PCA unified framework for data clustering," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 1, pp. 2–16, Jan. 2017.

[81] L. Vandenberghe and S. Boyd, "Semidefinite programming," *SIAM Rev.*, vol. 38, no. 1, pp. 49–95, Mar. 1996.

[82] G. H. Golub and C. Reinsch, "Singular value decomposition and least squares solutions," *Numerische Mathematik*, vol. 14, no. 5, pp. 403–420, 1970.

[83] A. Ahmed, N. Shervashidze, S. Narayanamurthy, V. Josifovski, and A. J. Smola, "Distributed large-scale natural graph factorization," in *Proc. 22nd Int. Conf. World Wide Web*, May 2013, pp. 37–48.

[84] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Y. Chang, "Network representation learning with rich text information," in *Proc. Int. Joint Conf. Artif. Intell.*, 2015, pp. 2111–2117.

[85] F. Xia, J. Liu, H. Nie, Y. Fu, L. Wan, and X. Kong, "Random walks: A review of algorithms and applications," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 4, no. 2, pp. 95–107, Nov. 2019.

[86] F. Xia, Z. Chen, W. Wang, J. Li, and L. T. Yang, "Mvcwalker: Random walk-based most valuable collaborators recommendation exploiting academic factors," *IEEE Trans. Emerg. Topics Comput.*, vol. 2, no. 3, pp. 364–375, Sep. 2014.

[87] M. A. Al-Garadi *et al.*, "Analysis of online social network connections for identification of influential users: Survey and open research issues," *ACM Comput. Surv.*, vol. 51, no. 1, pp. 1–37, 2018.

[88] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*., 2014, pp. 701–710.

[89] O. Levy and Y. Goldberg, "Neural word embedding as implicit matrix factorization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2177–2185.

[90] X. Rong, "word2vec parameter learning explained," 2014, *arXiv:1411.2738*.

[91] Y. Goldberg and O. Levy, "word2vec explained: Deriving Mikolov et al.'s negative-sampling word-embedding method,"2014, *arXiv:1402.3722*.

[92] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proc. 24th Int. Conf. World Wide Web*, 2015, pp. 1067–1077.

[93] W. Wang, J. Liu, Z. Yang, X. Kong, and F. Xia, "Sustainable collaborator recommendation based on conference closure," *IEEE Trans. Comput. Social Syst.*, vol. 6, no. 2, pp. 311–322, Apr. 2019.

[94] C. Tu, W. Zhang, Z. Liu, M. Sun *et al.* "Max-margin deepwalk: Discriminative learning of network representation." in *Proc. Int. Joint Conf. Artif. Intell.*, 2016, pp. 3889–3895.

[95] L. F. Ribeiro, P. H. Saverese, and D. R. Figueiredo, *"struc2vec: Learning node representations from structural identity*," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2017, pp. 385–394.

[96] Z. Yang, W. W. Cohen, and R. Salakhutdinov, "Revisiting semi-supervised learning with graph embeddings," in *Proc. 33rd Int. Conf. Mach. Learn.*, 2016, pp. 40–48.

[97] B. Adhikari, Y. Zhang, N. Ramakrishnan, and B. A. Prakash, "Distributed representations of subgraphs," in *Proc. IEEE Int. Conf. Data Mining Workshops*IEEE, 2017, pp. 111–117.

[98] A. Narayanan, M. Chandramohan, R. Venkatesan, L. Chen, Y. Liu, and S. Jaiswal, "graph2vec: Learning distributed representations of graphs," in *Proc. 13th Int. Workshop on Mining and Learning with Graphs (MLG)*, 2017.

[99] A. R. Benson, D. F. Gleich, and L.-H. Lim, "The spacey random walk: A stochastic process for higher-order data," *SIAM Rev.*, vol. 59, no. 2, pp. 321–345, 2017.

[100] H. Wang *et al.*, "Graphgan: Graph representation learning with generative adversarial nets," in *Proc. 32nd AAAI Conf. Artif. Intell.*, Apr. 2018, pp. 2508–2515.

[101] A. Bojchevski, O. Shchur, D. Zügner, and S. Günnemann, "Netgan: Generating graphs via random walks," *Proc. 35th Int. Conf. Mach. Learn.*, Jul. 2018, pp. 609–618.

[102] C. Shi, Y. Li, J. Zhang, Y. Sun, and S. Y. Philip, "A survey of heterogeneous information network analysis," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 1, pp. 17–37, Jan. 2017.

[103] N. Lao and W. W. Cohen, "Relational retrieval using a combination of path-constrained random walks," *Mach. Learn.*, vol. 81, no. 1, pp. 53–67, Oct. 2010.

[104] Q. Wang, Z. Mao, B. Wang, and L. Guo, "Knowledge graph embedding: A survey of approaches and applications," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 12, pp. 2724–2743, Sep. 2017.

[105] N. Lao, T. Mitchell, and W. W. Cohen, "Random walk inference and learning in a large scale knowledge base," in *Proc. Conf. Empirical Methods Natural Lang. Process. Assoc. Comput. Linguistics*, Jul. 2011, pp. 529–539.

[106] M. Gardner, P. P. Talukdar, B. Kisiel, and T. Mitchell, "Improving learning and inference in a large knowledge-base using latent syntactic cues," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2013, pp. 833–838.

[107] M. Gardner, P. Talukdar, J. Krishnamurthy, and T. Mitchell, "Incorporating vector space similarity in random walk inference over knowledge bases," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2014, pp. 397–406.

[108] W. Y. Wang and W. W. Cohen, "Joint information extraction and reasoning: A scalable statistical relational learning approach," in *Proc. 53rd Annu. Meeting Assoc. Comput. Linguistics 7th Int. Joint Conf. Natural Lang. Process.* (Volume 1: Long Papers), 2015, pp. 355–364.

[109] Q. Liu, L. Jiang, M. Han, Y. Liu, and Z. Qin, "Hierarchical random walk inference in knowledge graphs," in *Proc. 39th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2016, pp. 445–454.

[110] T.-y. Fu, W.-C. Lee, and Z. Lei, "Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning," in *Proc. ACM Conf. Inf. Knowl. Manage.*, 2017, pp. 1797–1806.

[111] Y. Dong, N. V. Chawla, and A. Swami, "metapath2vec: Scalable representation learning for heterogeneous networks," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2017, pp. 135–144.

[112] R. Hussein, D. Yang, and P. Cudré-Mauroux, "Are meta-paths necessary?: Revisiting heterogeneous graph embeddings," in *Proc. 27th ACM Int. Conf. Inf. Knowl. Manage.*, 2018, pp. 437–446.

[113] G. Wan, B. Du, S. Pan, and G. Haffari, "Reinforcement learning based meta-path discovery in large-scale heterogeneous information networks," in *Proc. AAAI Conf. Artif. Intell.*, Apr. 2020, pp. 6094–6101.

[114] C. Shi, B. Hu, W. X. Zhao, and S. Y. Philip, "Heterogeneous information network embedding for recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 2, pp. 357–370, Feb. 2019.

[115] J. Tang, M. Qu, and Q. Mei, "PTE: Predictive text embedding through large-scale heterogeneous text networks," in *Proc. 21th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2015, pp. 1165–1174.

[116] C. Zhang, A. Swami, and N. V. Chawla, "SHNE: Representation learning for semantic-associated heterogeneous networks," in *Proc. 12th ACM Int. Conf. Web Search Data Mining*, 2019, pp. 690–698.

[117] M. Hou, J. Ren, D. Zhang, X. Kong, D. Zhang, and F. Xia, "Network embedding: Taxonomies, frameworks and applications," *Comput. Sci. Rev.*, vol. 38, Nov. 2020, Art. no. 100296.

[118] G. H. Nguyen, J. B. Lee, R. A. Rossi, N. K. Ahmed, E. Koh, and S. Kim, "Continuous-time dynamic network embeddings," in *Proc. Companion Web Conf.*, 2018, pp. 969–976.

[119] Y. Zuo, G. Liu, H. Lin, J. Guo, X. Hu, and J. Wu, "Embedding temporal network via neighborhood formation," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, Jul. 2018, pp. 2857–2866.

[120] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1024–1034.

[121] M. Gori, G. Monfardini, and F. Scarselli, "A new model for learning in graph domains," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, 2005, pp. 729–734.

[122] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," in *2nd Int. Conf. Learn. Repres.*, Banff, AB, Canada 2014.

[123] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," Jun. 2015, *arXiv:1506.05163*.

[124] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 3844–3852.

[125] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *Proc. Int. Conf. Learn. Representations*, 2017.

[126] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model CNNs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 5115–5124.

[127] Z. Zhou and X. Li, "Graph convolution: A high-order and adaptive approach,"Jun. 2017, *arXiv:1706.09916*.

[128] F. Manessi, A. Rozza, and M. Manzo, "Dynamic graph convolutional networks," *Pattern Recognition*, vol. 97, p. 107000, 2020.

[129] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," *6th Int. Conf. Learn. Repres.*, Vancouver, BC, Canada, 2018.

[130] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," *Proc. 27th Int. Joint Conf. Artif. Intell.*, 2017, pp. 3634–3640.

[131] S. Yan, Y. Xiong, and D. Lin, "Spatial temporal graph convolutional networks for skeleton-based action recognition," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 3634–3640.

[132] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 2014–2023.

[133] D. K. Duvenaud *et al.*, "Convolutional networks on graphs for learning molecular fingerprints," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 2224–2232.

[134] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 1993–2001.

[135] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proc. 34th Int. Conf. Machine Learn*. 2017, pp. 1263–1272.

[136] M. Allamanis, M. Brockschmidt, and M. Khademi, "Learning to represent programs with graphs," in *Proc. Int. Conf. Learn. Representations*, 2018.

[137] C. Zhuang and Q. Ma, "Dual graph convolutional networks for graph-based semi-supervised classification," in *Proc. Web Conf.*, 2018, pp. 499–508.

[138] H. Dai, Z. Kozareva, B. Dai, A. Smola, and L. Song, "Learning steady-states of iterative algorithms over graphs," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1114–1122.

[139] S. Zhu, L. Zhou, S. Pan, C. Zhou, G. Yan, and B. Wang, "GSSNN: Graph smoothing splines neural networks," in *Proc. AAAI Conf. Artif. Intell.*, Apr. 2020, pp. 7007–7014.

[140] H. Gao, Z. Wang, and S. Ji, "Large-scale learnable graph convolutional networks," in *Proc. 24th ACM Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 1416–1424.

[141] C. Tran, W.-Y. Shin, A. Spitz, and M. Gertz, "DeepNC: Deep generative network completion," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2020, doi: 10.1109/TPAMI.2020.3032286.

[142] A. Vaswani *et al.*, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.

[143] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *Proc. Int. Conf. Learn. Representations*, 2018.
[144] J. Zhang, X. Shi, J. Xie, H. Ma, I. King, and D.-Y. Yeung, "GaAN: Gated attention networks for learning on large and spatiotemporal graphs," in *Proc. 34th Conf. Uncertainty Artif. Intell.*, Mar. 2018.
[145] J. B. Lee, R. Rossi, and X. Kong, "Graph classification using structural attention," in *Proc. 24th ACM Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 1666–1674.
[146] S. Abu-El-Haija, B. Perozzi, R. Al-Rfou, and A. A. Alemi, "Watch your step: Learning node embeddings via graph attention," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 9180–9190.
[147] M. Hou, L. Wang, J. Liu, X. Kong, and F. Xia, "A3graph: Adversarial attributed autoencoder for graph representation," in *Proc. 36th ACM Symp. Appl. Comput.*, 2021, pp. 1697–1704.
[148] S. Cao, W. Lu, and Q. Xu, "Deep neural networks for learning graph representations," in *Proc. 30th AAAI Conf. Artif. Intell.*, 2016, pp. 1145–1152.
[149] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2016, pp. 1225–1234.
[150] Y. Qi, Y. Wang, X. Zheng, and Z. Wu, "Robust feature learning by stacked autoencoder with maximum correntropy criterion," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2014, pp. 6716–6720.
[151] L. Jing and Y. Tian, "Self-supervised visual feature learning with deep neural networks: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, doi: 10.1109/TPAMI.2020.2992393, May 2020.
[152] K. Tu, P. Cui, X. Wang, P. S. Yu, and W. Zhu, "Deep recursive network embedding with regular equivalence," in *Proc. 24th ACM Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 2357–2366.
[153] T. N. Kipf and M. Welling, "Variational graph auto-encoders," 2016, *arXiv:1611.07308*.
[154] S. Pan, R. Hu, S.-f. Fung, G. Long, J. Jiang, and C. Zhang, "Learning graph embedding with adversarial training methods," *IEEE Trans. Cybern.*, vol. 50, no. 6, pp. 2475–2487, Sep. 2019.
[155] M. Schlichtkrull *et al.*, "Modeling Relational Data With Graph Convolutional Networks," in *Proc. Eur. Semantic Web Conf.*, 2018, pp. 593–607.
[156] Y. Li, O. Vinyals, C. Dyer, R. Pascanu, and P. Battaglia, "Learning deep generative models of graphs,"2018, *arXiv:1803.03324*.
[157] Y. Xian, C. H. Lampert, B. Schiele, and Z. Akata, "Zero-shot learning-a comprehensive evaluation of the good, the bad and the ugly," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 9, pp. 2251–2265, Sep. 2018.
[158] M. R. Vyas, H. Venkateswara, and S. Panchanathan, "Leveraging seen and unseen semantic relationships for generative zero-shot learning," in *Proc. Eur. Conf. Comput. Vis*, 2020, pp. 70–86.
[159] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, "Graph wavenet for deep spatial-temporal graph modeling," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, 2019, pp. 1907–1913.
[160] L. Yao, C. Mao, and Y. Luo, "Graph convolutional networks for text classification," in *Proc. AAAI Conf. Artif. Intell.*,2019, pp. 7370–7377.
[161] Y. Zhang, Q. Liu, and L. Song, "Sentence-state LSTM for text representation," in *Proc. 56th Annu. Meeting Assoc. Comput. Linguistics*, 2018, pp. 317–327.
[162] D. Marcheggiani and I. Titov, "Encoding sentences with graph convolutional networks for semantic role labeling," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2017, pp. 1506–1515.
[163] D. Beck, G. Haffari, and T. Cohn, "Graph-to-sequence learning using gated graph neural networks," in *Proc. 56th Annu. Meeting Assoc. Comput. Linguistics* (Volume 1: Long Papers), 2018, pp. 273–283.
[164] H. Peng *et al.*, "Large-scale hierarchical text classification with recursively regularized deep graph-CNN," in *Proc. Web Conf.*, 2018, pp. 1063–1072.
[165] Z. Wang, T. Chen, J. Ren, W. Yu, H. Cheng, and L. Lin, "Deep reasoning with knowledge graph for social relationship understanding," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, 2018, pp. 1021–1028.
[166] C.-W. Lee, W. Fang, C.-K. Yeh, and Y.-C. Frank Wang, "Multi-label zero-shot learning with structured knowledge graphs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 1576–1585.
[167] P. Battaglia, R. Pascanu, M. Lai, D. J. Rezende, "Interaction networks for learning about objects, relations and physics," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 4502–4510.
[168] N. Watters, D. Zoran, T. Weber, P. Battaglia, R. Pascanu, and A. Tacchetti, "Visual interaction networks: Learning a physics simulator from video," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 4539–4547.
[169] K. T. Butler, D. W. Davies, H. Cartwright, O. Isayev, and A. Walsh, "Machine learning for molecular and materials science," *Nature*, vol. 559, no. 7715, pp. 547–555, 2018.
[170] S. Rhee, S. Seo, and S. Kim, "Hybrid approach of relation network and localized graph convolutional filtering for breast cancer subtype classification," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, 2018, pp. 3527–3534.
[171] S. Ji, S. Pan, E. Cambria, P. Marttinen, and P. S. Yu, "A survey on knowledge graphs: Representation, acquisition and applications," 2020, *arXiv:2002.00388*.
[172] T. Hamaguchi, H. Oiwa, M. Shimbo, and Y. Matsumoto, "Knowledge base completion with out-of-knowledge-base entities: A graph neural network approach," *Trans. Japanese Soc. Artif. Intell.*, vol. 33, pp. 1–10, 2018.
[173] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 2787–2795.
[174] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes," in *Proc. 28th AAAI Conf. Artif. Intell.*, 2014, pp. 1112–1119.
[175] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion," in *Proc. 29th AAAI Conf. Artif. Intell.*, 2015, pp. 2181–2187.
[176] G. Ji, S. He, L. Xu, K. Liu, and J. Zhao, "Knowledge graph embedding via dynamic mapping matrix," in *Proc. 53rd Annu. Meeting Assoc. Comput. Linguistics 7th Int. Joint Conf. Natural Lang. Process.* (Volume 1: Long Papers), vol. 1, 2015, pp. 687–696.
[177] J. Feng, M. Huang, M. Wang, M. Zhou, Y. Hao, and X. Zhu, "Knowledge graph embedding by flexible translation," in *Proc. 15th Int. Conf. Princ. Knowl. Representation Reasoning*, 2016, pp. 557–560.
[178] Z. Huang and N. Mamoulis, "Heterogeneous information network embedding for meta path based proximity," 2017, *arXiv:1701.05291*.
[179] R. Jenatton, N. L. Roux, A. Bordes, and G. R. Obozinski, "A latent factor model for highly multi-relational data," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 3167–3175.
[180] B. Yang, W.-t. Yih, X. He, J. Gao, and L. Deng, "Embedding entities and relations for learning and inference in knowledge bases," in *Proc. Int. Conf. Learn. Representations*, 2015.
[181] H. Liu, Y. Wu, and Y. Yang, "Analogical inference for multi-relational embeddings," in *Proc. 34th Int. Conf. Machine Learn. Org.*, 2017, pp. 2168–2178.
[182] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, "Neural combinatorial optimization with reinforcement learning," in *Proc. Int. Conf. Learn. Representations*, 2017.
[183] E. Khalil, H. Dai, Y. Zhang, B. Dilkina, and L. Song, "Learning combinatorial optimization algorithms over graphs," in *Adv. Neural Inf. Process. Syst.*, 2017, pp. 6348–6358.
[184] A. Nowak, S. Villar, A. S. Bandeira, and J. Bruna, "Revised note on learning quadratic assignment with graph neural networks," in *Proc. IEEE Data Sci. Workshop*, 2018, pp. 1–5.

Feng Xia (Senior Member, IEEE) received the B.Sc. and Ph.D. degrees in automation from Zhejiang University, Hangzhou, China, in 2001 and 2006, respectively.

He is currently an Associate Professor and Discipline Leader with the School of Engineering, IT and Physical Sciences, Federation University Australia, Ballarat, VIC, Australia. He has authored or coauthored two books and over 300 scientific papers in international journals and conferences. His research interests include data science, computational intelligence, social computing, and systems engineering.

Dr. Xia is a Senior Member of ACM.

Ke Sun received the B.Sc. and M.Sc. degrees in computer science and technology from Shandong Normal University, Jinan, China, in 2012 and 2015. He is currently working toward the Ph.D. degree in software engineering with the Dalian University of Technology, Dalian, China.

His research interests include deep learning, network representation learning, and knowledge graph.

Shuo Yu (Member, IEEE) received the B.Sc. degree in information and computing science and M.Sc. degree in applied mathematics from the Shenyang University of Technology, Shenyang, China, in 2011 and 2014, respectively, and the Ph.D. degree in software engineering from the Dalian University of Technology, Dalian, China, in 2019.

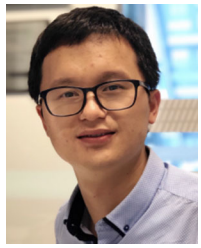
She is currently a Postdoctoral Research Fellow with the School of Software, Dalian University of Technology. She has authored or coauthored over 30 papers in ACM/IEEE conferences, journals, and magazines. Her research interests include network science, data science, and computational social science.

Abdul Aziz received the bachelor's degree in computer science from the COMSATS Institute of Information Technology, Lahore, Pakistan, in 2013, and the master's degree in computer science from the National University of Computer and Emerging Sciences, Karachi, Pakistan, in 2018. He is currently working toward the Ph.D. degree in software engineering with Alpha Lab, Dalian University of Technology, Dalian, China.

His research interests include big data, information retrieval, graph learning, and social computing.

Liangtian Wan (Member, IEEE) received the B.S. degree in electronic information engineering and the Ph.D. degree in information and communication engineering from Harbin Engineering University, Harbin, China, in 2011 and 2015, respectively.

From October 2015 to April 2017, he has been a Research Fellow with Nanyang Technological University, Singapore. He is currently an Associate Professor with the School of Software, Dalian University of Technology, Dalian, China. He is the author of over 70 papers. His current research interests include data science, big data, and graph learning.



Shirui Pan received the Ph.D. degree in computer science from the University of Technology Sydney (UTS), Ultimo NSW, Australia, in 2015.

He is currently a Lecturer with the Faculty of Information Technology, Monash University, Melbourne, VIC, Australia. He has authored or coauthored over 60 research papers in top-tier journals and conferences. His research interests include data mining and machine learning.

Huan Liu (Fellow, IEEE) received the B.Eng. degree in computer science and electrical engineering from Shanghai Jiaotong University, Shanghai, China, in 1983, and the Ph.D. degree in computer science from the University of Southern California, Los Angeles, CA, USA, in 1989.

He is currently a Professor of Computer Science and Engineering with Arizona State University, Tempe, AZ, USA. His well-cited publications include books, book chapters, and encyclopedia entries and conference, and journal papers. His research interests include data mining, machine learning, social computing, and artificial intelligence, investigating problems that arise in many real-world applications with high-dimensional data of disparate forms.

Dr. Liu is also a fellow of ACM, AAAI, and AAAS.