Chapter 3
Graph Neural Networks

Lingfei Wu, Peng Cui, Jian Pei, Liang Zhao and Le Song

Abstract Deep Learning has become one of the most dominant approaches in Artificial Intelligence research today. Although conventional deep learning techniques have achieved huge successes on Euclidean data such as images, or sequence data such as text, there are many applications that are naturally or best represented with a graph structure. This gap has driven a tide in research for deep learning on graphs, among them Graph Neural Networks (GNNs) are the most successful in coping with various learning tasks across a large number of application domains. In this chapter, we will systematically organize the existing research of GNNs along three axes: foundations, frontiers, and applications. We will introduce the fundamental aspects of GNNs ranging from the popular models and their expressive powers, to the scalability, interpretability and robustness of GNNs. Then, we will discuss various frontier research, ranging from graph classification and link prediction, to graph generation and transformation, graph matching and graph structure learning. Based on them, we further summarize the basic procedures which exploit full use of various GNNs for a large number of applications. Finally, we provide the organization of our book and summarize the roadmap of the various research topics of GNNs.

Lingfei Wu
JD.COM Silicon Valley Research Center, e-mail: lwu@email.wm.edu

Peng Cui
Department of Computer Science, Tsinghua University, e-mail: cuip@tsinghua.edu.cn

Jian Pei
Department of Computer Science, Simon Fraser University, e-mail: jpei@cs.sfu.ca

Liang Zhao
Department of Computer Science, Emory University, e-mail: liang.zhao@emory.edu

Le Song
Mohamed bin Zayed University of Artificial Intelligence, e-mail: dasongle@gmail.com

L. Wu et al. (eds.), *Graph Neural Networks: Foundations, Frontiers, and Applications*,
https://doi.org/10.1007/978-981-16-6054-2_3

3.1 Graph Neural Networks: An Introduction

Deep Learning has become one of the most dominant approaches in Artificial Intelligence research today. Conventional deep learning techniques, such as recurrent neural networks (Schuster and Paliwal, 1997) and convolutional neural networks (Krizhevsky et al, 2012) have achieved huge successes on Euclidean data such as images, or sequence data such as text and signals. However, in a rich variety of scientific fields, many important real-world objects and problems can be naturally or best expressed along with a complex structure, e.g., graph or manifold structure, such as social networks, recommendation systems, drug discovery and program analysis. On the one hand, these graph-structured data can encode complicated pairwise relationships for learning more informative representations; On the other hand, the structural and semantic information in original data (images or sequential texts) can be exploited to incorporate domain-specific knowledge for capturing more fine-grained relationships among the data.

In recent years, deep learning on graphs has experienced a burgeoning interest from the research community (Cui et al, 2018; Wu et al, 2019e; Zhang et al, 2020e). Among them, Graph Neural Networks (GNNs) is the most successful learning framework in coping with various tasks across a large number of application domains. Newly proposed neural network architectures on graph-structured data (Kipf and Welling, 2017a; Petar et al, 2018; Hamilton et al, 2017b) have achieved remarkable performance in some well-known domains such as social networks and bioinformatics. They have also infiltrated other fields of scientific research, including recommendation systems (Wang et al, 2019j), computer vision (Yang et al, 2019g), natural language processing (Chen et al, 2020o), program analysis (Allamanis et al, 2018b), software mining (LeClair et al, 2020), drug discovery (Ma et al, 2018), anomaly detection (Markovitz et al, 2020), and urban intelligence (Yu et al, 2018a).

Despite these successes that existing research has achieved, GNNs still face many challenges when they are used to model highly-structured data that is time-evolving, multi-relational, and multi-modal. It is also very difficult to model mapping between graphs and other highly structured data, such as sequences, trees, and graphs. One challenge with graph-structured data is that it does not show as much spatial locality and structure as image or text data does. Thus, graph-structured data is not naturally suitable for highly regularized neural structures such as convolutional and recurrent neural networks.

More importantly, new application domains for GNNs that emerge from real-world problems introduce significantly challenges for GNNs. Graphs provide a powerful abstraction that can be used to encode arbitrary data types such as multidimensional data. For example, similarity graphs, kernel matrices, and collaborative filtering matrices can also be viewed as special cases of graph structures. Therefore, a successful modeling process of graphs is likely to subsume many applications that are often used in conjunction with specialized and hand-crafted methods.

In this chapter, we will systematically organize the existing research of GNNs along three axes: foundations of GNNs, frontiers of GNNs, and GNN based applications. First of all, we will introduce the fundamental aspects of GNNs ranging from

popular GNN methods and their expressive powers, to the scalability, interpretability, and robustness of GNNs. Next, we will discuss various frontier research which are built on GNNs, including graph classification, link prediction, graph generation and transformation, graph matching, graph structure learning, dynamic GNNs, heterogeneous GNNs, AutoML of GNNs and self-supervised GNNs. Based on them, we further summarize the basic procedures which exploit full use of various GNNs for a large number of applications. Finally, we provide the organization of our GNN book and summarize the roadmap of the various research topics of GNNs.

3.2 Graph Neural Networks: Overview

In this section, we summarize the development of graph neural networks along three important dimensions: (1) Foundations of GNNs; (2) Frontiers of GNNs; (3) GNN-based applications. We will first discuss the important research areas under the first two dimensions for GNNs and briefly illustrate the current progress and challenges for each research sub-domain. Then we will provide a general summarization on how to exploit the power of GNNs for a rich variety of applications.

3.2.1 Graph Neural Networks: Foundations

Conceptually, we can categorize the fundamental learning tasks of GNNs into five different directions: i) Graph Neural Networks Methods; ii) Theoretical understanding of Graph Neural Networks; iii) Scalability of Graph Neural Networks; iv) Interpretability of Graph Neural Networks; and v) Adversarial robustness of Graph Neural Networks. We will discuss these fundamental aspects of GNNs one by one in this subsection.

Graph Neural Network Methods. Graph Neural Networks are specifically designed neural architectures operated on graph-structure data. The goal of GNNs is to iteratively update the node representations by aggregating the representations of node neighbors and their own representation in the previous iteration. There are a variety of graph neural networks proposed in the literature (Kipf and Welling, 2017a; Petar et al, 2018; Hamilton et al, 2017b; Gilmer et al, 2017; Xu et al, 2019d; Velickovic et al, 2019; Kipf and Welling, 2016), which can be further categorized into supervised GNNs and unsupervised GNNs. Once the node representations are learnt, a fundamental task on graphs is node classification that tries to classify the nodes into a few predefined classes. Despite the huge successes that various GNNs have achieved, a severe issue on training deep graph neural networks has been observed to yield inferior results, namely, over-smoothing problem (Li et al, 2018b), where all the nodes have similar representations. Many recent works have been proposed with different remedies to overcome this over-smoothing issue.

Theoretical understanding of Graph Neural Networks. Rapid algorithmic developments of GNNs have aroused a significant amount of interests in theoretical analysis on the expressive power of GNNs. In particular, much efforts have been made in order to characterize the expressive power of GNNs when compared with the traditional graph algorithms (e.g. graph kernel-based methods) and how to build more powerful GNNs so as to overcome several limitations in GNNs. Specifically, Xu et al (2019d) showed that current GNN methods are able to achieve the expressive power of the 1-dimensional Weisfeiler-Lehman test (Weisfeiler and Leman., 1968), a widely used method in traditional graph kernel community (Shervashidze et al, 2011b). Much recent research has further proposed a series of design strategies in order to further reach beyond the expressive power of the Weisfeiler-Lehman test by including attaching random attributes, distance attributes, and utilizing higher-order structures.

Scalability of Graph Neural Networks. The increasing popularity of GNNs have attracted many attempts to apply various GNN methods on real-world applications, where the graph sizes are often about having one hundred million nodes and one billion edges. Unfortunately, most of the GNN methods cannot directly be applied on these large-scale graph-structured data due to large memory requirements (Hu et al, 2020b). Specifically, this is because the majority of GNNs are required to store the whole adjacent matrices and the intermediate feature matrices in the memory, rendering the significant challenges for both computer memory consumption and computational costs. In order to address these issues, many recent works have been proposed with various sampling strategies such as node-wise sampling (Hamilton et al, 2017b; Chen et al, 2018d), layer-wise sampling (Chen and Bansal, 2018; Huang et al, 2018), and graph-wise sampling (Chiang et al, 2019; Zeng et al, 2020a).

Interpretability of Graph Neural Networks. Explainable artificial intelligence are becoming increasingly popular in providing interpretable results on machine learning process, especially due to the black-box issue of deep learning techniques. As a result, there is a surge of interests in improving the interpretability of GNNs. Generally speaking, explanation results on GNNs could be important nodes, important edges, or important features of nodes or edges. Technically, white-box approximation based methods (Baldassarre and Azizpour, 2019; Sanchez-Lengeling et al, 2020) utilize the information inside the model inlucidng gradients, intermediate features, and model parameters to provide the explanation. In contrast, the black-box approximation based methods (Huang et al, 2020c; Zhang et al, 2020a; Vu and Thai, 2020) abandon the utilization of internal information of complex models but instead leverage the intrinsically interpretable simple models (e.g. linear regression and decision trees) to fit the complex models. However, most of the existing works are time-consuming, which rendering the difficulty in coping with large-scale graph. To this end, many recent efforts have been made in order to develop more efficient approaches without compromising the explanation accuracy.

Adversarial robustness of Graph Neural Networks. Trustworthy machine learning has recently attracted a significant amount of attention since the existing studies have shown that deep learning models could be deliberately fooled, evaded, misled, and stolen (Goodfellow et al, 2015). Consequently, a line of research has exten-

sively studied the robustness of models in domains like computer vision and natural language processing, which has also influenced similar research on the robustness of GNNs. Technically, the standard approach (via adversarial examples) for studying the robustness of GNNs is to construct a small change of the input graph data and then to observe if it leads to a large change of the prediction results (i.e. node classification accuracy). There are a growing number of research works toward either adversarial attacks (Dai et al, 2018a; Wang and Gong, 2019; Wu et al, 2019b; Zügner et al, 2018; Zügner et al, 2020) or adversarial training (Xu et al, 2019c; Feng et al, 2019b; Chen et al, 2020i; Jin and Zhang, 2019). Many recent efforts have been made to provide both theoretical guarantees and new algorithmic developments in adversarial training and certified robustness.

3.2.2 Graph Neural Networks: Frontiers

Built on these aforementioned fundamental techniques of GNNs, there are various fast-growing recent research developments in coping with a variety of graph-related research problems. In this section, we will comprehensively introduce these research frontiers that are either long-standing graph learning problems with new GNN solutions or recently emerging learning problems with GNNs.

Graph Neural Networks: Graph Classification and Link Prediction. Since each layer in GNN models only produce the node-level representations, graph pooling layers are needed to further compute graph-level representation based on node-level representations. The graph-level representation, which summarizes the key characteristics of input graph-structure, is the critical component for the graph classification. Depending on the learning techniques of graph pooling layers, these methods can be generally categorized into four groups: simple flat-pooling (Duvenaud et al, 2015a; Mesquita et al, 2020), attention-based pooling (Lee et al, 2019d; Huang et al, 2019d), cluster-based pooling (Ying et al, 2018c), and other type of pooling (Zhang et al, 2018f; Bianchi et al, 2020; Morris et al, 2020b). Beside graph classification, another long-standing graph learning problem is link prediction task, which aims to predict missing or future links between any pair of nodes. Since GNNs can jointly learn from both graph structure and side information (e.g. node and edge features), it has shown great advantages over other conventional graph learning methods for link prediction. Regarding the learning types of link prediction, node-based methods (Kipf and Welling, 2016) and subgraph-based methods (Zhang and Chen, 2018a, 2020) are two popular groups of GNN based methods.

Graph Neural Networks: Graph Generation and Graph Transformation. Graph generation problem that builds probabilistic models over graphs is a classical research problem that lies at the intersection between the probability theory and the graph theory. Recent years have seen an increasing amount of interest in developing deep graph generative models that are built on modern deep learning on graphs techniques like GNNs. These deep models have proven to be a more successful approach in capturing the complex dependencies within the graph data and generating

more realistic graphs. Encouraged by the great successes of Variational AutoEncoder (VAE) (Kingma and Welling, 2013) and Generative Adversarial Networks (Goodfellow et al, 2014a) (Goodfellow et al, 2014b), there are three representative GNN based learning paradigms for graph generation including GraphVAE approaches (Jin et al, 2018b; Simonovsky and Komodakis, 2018; Grover et al, 2019), GraphGAN approaches (De Cao and Kipf, 2018; You et al, 2018a) and Deep Autoregressive methods (Li et al, 2018d; You et al, 2018b; Liao et al, 2019a). Graph transformation problem can be formulated as a conditional graph generation problem, where its goal is to learn a translation mapping between the input source graph and the output target graph (Guo et al, 2018b). Such learning problem often arises in other domains such as machine translation problem in Natural Language Processing domain and image style transfer in computer Vision domain. Depending on what graph information is transformed, this problem can be generally grouped into four categories including node-level transformation (Battaglia et al, 2016; Yu et al, 2018a; Li et al, 2018e), edge-level transformation (Guo et al, 2018b; Zhu et al, 2017; Do et al, 2019), node-edge co-transformation (Maziarka et al, 2020a; Kaluza et al, 2018; Guo et al, 2019c), and graph-involved transformation (Bastings et al, 2017; Xu et al, 2018c; Li et al, 2020f).

Graph Neural Networks: Graph Matching and Graph Structure Learning. The problem of graph matching is to find the correspondence between two input graphs, which is an extensively studied problem in a variety of research fields. Conventionally, the graph matching problem is known to be NP-hard (Loiola et al, 2007), rendering this problem computationally infeasible for exact and optimum solutions for real-world large-scale problems. Due to the expressive power of GNNs, there is an increasing attention on developing various graph matching methods based on GNNs in order to improve the matching accuracy and efficiency (Zanfir and Sminchisescu, 2018; Rolínek et al, 2020; Li et al, 2019h; Ling et al, 2020). Graph matching problem aims to measure the similarity between two graph structures without changing them. In contrast, graph structure learning aims to produce an optimized graph structure by jointly learning implicit graph structure and graph node representation (Chen et al, 2020m; Franceschi et al, 2019; Velickovic et al, 2020). The learnt graph structure often can be treated as a shift compared to the intrinsic graph which is often noisy or incomplete. Graph structure learning can also be used when the initial graph is not provided while the data matrix shows correlation among data points.

Dynamic Graph Neural Networks and Heterogeneous Graph Neural Networks. In real-world applications, the graph nodes (entities) and the graph edges (relations) are often evolving over time, which naturally gives rise to dynamic graphs. Unfortunately, various GNNs cannot be directly applied to the dynamic graphs, where modeling the evolution of the graph is critical in making accurate predictions. A simple yet often effective approach is converting dynamic graphs into static graphs, leading to potential loss of information. Regarding the type of dynamic graphs, there are two major categories of GNN-based methods, including GNNs for discrete-time dynamic graphs (Seo et al, 2018; Manessi et al, 2020) and GNNs for continue-time dynamic graphs (Kazemi et al, 2019; Xu et al, 2020a). Independently, another pop-

ular graph type in real applications is heterogeneous graphs that consist of different types of graph nodes and edges. To fully exploit this information in heterogeneous graphs, different GNNs for homogeneous graphs are not applicable. As a result, a new line of research has been devoted to developing various heterogeneous graph neural networks including message passing based methods (Wang et al, 2019l; Fu et al, 2020; Hong et al, 2020b), encoder-decoder based methods (Tu et al, 2018; Zhang et al, 2019b), and adversarial based methods (Wang et al, 2018a; Hu et al, 2018a).

Graph Neural Networks: AutoML and Self-supervised Learning. Automated machine learning (AutoML) has recently drawn a significant amount of attention in both research and industrial communities, the goal of which is coping with the huge challenge of time-consuming manual tuning process, especially for complicated deep learning models. This wave of the research in AutoML also influences the research efforts in automatically identifying an optimized GNN model architecture and training hyperparameters. Most of the existing research focuses on either architecture search space (Gao et al, 2020b; Zhou et al, 2019a) or training hyperparameter search space (You et al, 2020a; Shi et al, 2020). Another important research direction of GNNs is to address the limitation of most of deep learning models that requires large amount of annotated data. As a result, self-supervised learning has been proposed which aims to design and leverage domain-specific pretext tasks on unlabeled data to pretrain a GNN model. In order to study the power of serf-supervised leanring in GNNs, there are quite a few works that systemmatically design and compare different self-supervised pretext tasks in GNNs (Hu et al, 2020c; Jin et al, 2020d; You et al, 2020c).

3.2.3 Graph Neural Networks: Applications

Due to the power of GNNs to model various data with complex structures, GNNs have been widely applied into many applications and domains, such as modern recommender systems, computer vision (CV), natural language processing (NLP), program analysis, software mining, bioinformatics, anomaly detection, and urban intelligence. Though GNNs are utilized to solve different tasks for different applications, they all consist of two important steps, namely graph construction and graph representation learning. Graph construction aims to first transform or represent the input data as graph-structured data. Based on the graphs, graph representation learning utilizes GNNs to learn the node or graph embeddings for the downstream tasks. In the following, we briefly introduce the techniques of these two steps regarding different applications.

3.2.3.1 Graph Construction

Graph construction is important in capturing the dependency among the objects in the input data. Given the various formats of input data, different applications have different graph construction techniques, while some tasks need to pre-define the semantic meaning of nodes and edges to fully express the structural information of the input data.

Input Data with Explicit Graph Structures. Some applications naturally have the structure inside the data without pre-defined nodes and the edges/relationships among them. For example, the user-item interactions in a recommender systems naturally form a graph where user-item preference is regarded as the edges between the nodes of user and item. In the task of drug design, a molecule is also naturally represented as a graph, where each node denotes an atom and an edge denotes a bond that connects two atoms. In the task of protein function prediction and interaction, the graph can also easily fit into a protein, where each amino-acid refers to a node and each edge refers to the interaction among amino-acids.

Some graphs are constructed with the node and edge attributes. For example, in dealing with the transportation in the urban intelligence, the traffic networks can be formalized as an undirected graph to predict the traffic state. Specifically, the nodes are the traffic sensing locations, e.g., sensor stations, road segments, and the edges are the intersections or road segments connecting those traffic sensing locations. Some urban traffic network can be modeled as a directed graph with attributes to predict the traffic speed, where the nodes are the road segments, and the edges are the intersections. Road segment width, length, and direction are the attributes of the nodes, and the type of intersection, and whether there are traffic lights, toll gates are the attributes of edges.

Input Data with Implicit Graph Structures. For many tasks that do not naturally involve a structured data, graph construction becomes very challenging. It is important to choose the best representation so that the nodes and edges can capture all the important things. For example, in computer vision (CV) tasks, there are three kinds of graph construction. The first is to split the image or the frame of the video into regular grids, and each grid serves as a vertex of the visual graph. The second way is to first get the preprocessed structures which can be directly borrowed for vertex representation, such as the formulation of scene graphs. The last one is about utilizing semantic information to represent visual vertexes, such as assigning pixels with similar features to the same vertex. The edges in the visual images can capture two kinds of information. One is spatial information. For example, for static methods, generating scene graphs (Xu et al, 2017a) and human skeletons (Jain et al, 2016a) is natural to choose edges between nodes in the visual graph to represent their location connection. Another is temporal information. For example, to represent the video, the model not only builds spatial relations in a frame but also captures temporal connections among adjacent frames.

In the natural language processing (NLP) tasks, the graph construction from the text data can be categorized into five categories: text graphs, syntactic graphs, semantic graphs, knowledge graphs, and hybrid graphs. Text graphs normally re-

gard words, sentences, paragraphs, or documents as nodes and establish edges by word co-occurrence, location, or text similarities. Syntactic graphs (or trees) emphasize the syntactical dependencies between words in a sentence, such as dependency graph and constituency graph. Knowledge graphs (KGs) are graphs of data intended to accumulate and convey knowledge of the real world. Hybrid graphs contain multiple types of nodes and edges to integrate heterogeneous information. In the task of program analysis, the formulation over graph representations of programs includes syntax trees, control flow, data flow, program dependence, and call graphs, each providing different views of a program. At a high level, programs can be thought as a set of heterogeneous entities that are related through various kinds of relations. This view directly maps a program to a heterogeneous directed graph, with each entity being represented as a node and each relationship of type represented as an edge.

3.2.3.2 Graph Representation Learning

After getting the graph expression of the input data, the next step is applying GNNs for learning the graph representations. Some works directly utilize the typical GNNs, such as GCN (Kipf and Welling, 2017a), GAT (Petar et al, 2018), GGNN (Li et al, 2016a) and GraphSage (Hamilton et al, 2017b), which can be generalized to different application tasks. While some special tasks needs an additional design on the GNN architecture to better handle the specific problem. For example, in the task of recommender systems, PinSage (Ying et al, 2018a) is proposed which takes the top-k counted nodes of a node as its receptive field and utilizes weighted aggregation for aggregation. PinSage can be scalable to the web-scale recommender systems with millions of users and items. KGCN (Wang et al, 2019d) aims to enhance the item representation by performing aggregations among its corresponding entity neighborhood in a knowledge graph. KGAT (Wang et al, 2019j) shares a generally similar idea with KGCN except for incorporating an auxiliary loss for knowledge graph reconstruction. For instance, in the NLP task of KB-alignment, Xu et al (2019e) formulated it as a graph matching problem, and proposed a graph attention-based approach. It first matches all entities in two KGs, and then jointly models the local matching information to derive a graph-level matching vector. The detailed GNN techniques for each application can be found in the following chapters of this book.

3.2.4 Graph Neural Networks: Organization

The high-level organization of the book is demonstrated in Figure 1.3. The book is organized into four parts to best accommodate a variety of readers. Part I introduces basic concepts; Part II discusses the most established methods; Part III presents the most typical frontiers, and Part IV describes advances of methods and applications

that tend to be important and promising for future research. Next, we briefly elaborate on each chapter.

- *Part I: Introduction*. These chapters provide the general introduction from the representation learning for different data types, to the graph representation learning. In addition, it introduces the basic ideas and typical variants of graph neural networks for the graph representation learning.
- *Part II: Foundations*. These chapters describe the foundations of the graph neural networks by introducing the properties of graph neural networks as well as several fundamental problems in this line. Specifically, this part introduces the fundamental problems in graphs: node classification, the expressive power of graph neural networks, the interpretability and scalability issues of graph neural network, and the adversarial robustness of the graph neural networks.
- *Part III: Frontiers*. In these chapters, some frontier or advanced problems in the domain of graph neural networks are proposed. Specifically, there are introductions about the techniques in graph classification, link prediction, graph generation, graph transformation, graph matching, graph structure learning. In addition, there are also introductions of several variants of GNNs for different types of graphs, such as GNNs for dynamic graphs, heterogeneous graphs. We also introduce the AutoML and self-supervised learning for GNNs.
- *Part IV: Broad and Emerging Applications*. These chapters introduce the broad and emerging applications with GNNs. Specifically, these GNNs-based applications covers modern recommender systems, tasks in computer vision and NLP, program analysis, software mining, biomedical knowledge graph mining for drug design, protein function prediction and interaction, anomaly detection, and urban intelligence.

3.3 Summary

Graph Neural Networks (GNNs) have been emerging rapidly to deal with the graph-structured data, which cannot be directly modeled by the conventional deep learning techniques that are designed for Euclidean data such as images and text. A wide range of applications can be naturally or best represented with graph structure and have been successfully handled by various graph neural networks.

In this chapter, we have systematically introduced the development and overview of GNNs, including the introduction of its foundations, frontiers, and applications. Specifically, we provide the fundamental aspects of GNNs ranging from the existing typical GNN methods and their expressive powers, to the scalability, interpretability and robustness of GNNs. These aspects motivate the research on better understanding and utilization of GNNs. Built on GNNs, recent research developments have seen a surge of interests in coping with graph-related research problems, which we called frontiers of GNNs. We have discussed various frontier research built on GNNs, ranging from graph classification and link prediction, to graph generation,

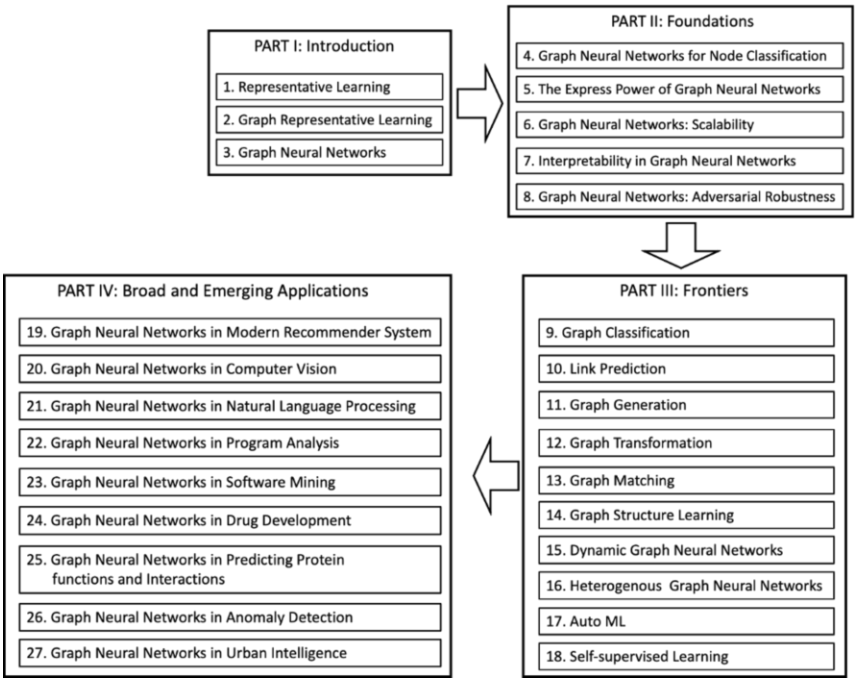

Fig. 3.1: The high-level organization of the book

transformation, matching and graph structure learning. Due to the power of GNNs to model various data with complex structures, GNNs have been widely applied into many applications and domains, such as modern recommender systems, computer vision, natural language processing, program analysis, software mining, bioinformatics, anomaly detection, and urban intelligence. Most of these tasks consist of two important steps, namely graph construction and graph representation learning. Thus, we provide the introduction of the techniques of these two steps regarding different applications. The introduction part will end here and thus a summary of the organization of this book has been provided at the end of this chapter.