大规模图神经网络研究综述

肖国庆[1),2)] 李雪琪[1)] 陈玥丹[1),2)] 唐 卓[1)] 姜文君[1)] 李肯立[1)]

[1)](湖南大学信息科学与工程学院 长沙 410082)

[2)](湖南大学深圳研究院 广东 深圳 518000)

摘 要 图神经网络凭借其处理非欧氏空间数据及其复杂特征方面的优越性受到了大量的关注,并且被广泛应用于推荐系统、知识图谱、交通道路分析等场景中.面对大规模数据,图结构的不规则性、节点特征的复杂性以及训练样本之间的依赖性对图神经网络模型的计算效率、内存管理以及分布式系统中的通信开销造成了巨大的压力.为应对和缓解以上问题,研究者从应用场景、算法模型、编程框架和硬件结构等多个层面对其进行了优化.本文主要回顾和总结了算法模型及编程框架方面的优化,为读者了解面向大规模数据的图神经网络采样算法以及框架优化相关工作提供帮助,为未来算法-框架协同优化奠定基础.具体来说,本文首先简要介绍图神经网络模型中的消息传递机制,分类介绍常见的图神经网络模型,并分析其在大规模数据训练中面临的困难和挑战;然后对面向大规模数据的图神经网络算法模型进行分类总结和分析,包括基于节点、边和子图的采样算法;接着介绍图神经网络编程框架加速的相关进展,主要包括主流框架的介绍以及优化技术的分类总结和分析;最后对未来面向大规模数据的图神经网络研究进行展望.

关键词 图神经网络;大规模数据;算法优化;框架加速

中图法分类号 TP391 **DOI 号** 10.11897/SP.J.1016.2024.00148

A Survey of Large-Scale Graph Neural Networks

XIAO Guo-Qing[1),2)] LI Xue-Qi[1)] CHEN Yue-Dan[1),2)] TANG Zhuo[1)] JIANG Wen-Jun[1)] LI Ken-Li[1)]

[1)](*College of Computer Science and Electronic Engineering, Hunan University, Changsha* 410082)

[2)](*Shenzhen Research Institute, Hunan University, Shenzhen, Guangdong* 518000)

Abstract Graph Neural Networks (GNNs) have garnered increasing attention for their ability to model non-Euclidean graph structures and complex features. They have been applied extensively in various application domains, such as recommender systems, link prediction, and traffic prediction. However, training GNN models on large-scale data poses several challenges, such as irregular graph structures, complex node features, and dependent graph training samples. These challenges can put a strain on computation efficiency, memory management, and the communication cost of distributed computing. To overcome these challenges, many researchers have focused on optimizing application methods, algorithm models, programming frameworks, and hardware design. This survey specifically focuses on algorithm optimization and framework acceleration for large-scale GNN models. By examining related works in these areas, this survey aims to help readers understand the existing research as well as lay the foundation for co-optimizing GNN algorithms and frameworks for large-scale data. This survey is structured as follows. Firstly, we provide an overview of the challenges faced by GNNs in large-scale applications and the major optimization

收稿日期:2022-06-05;在线发布日期:2023-04-19.本课题得到广东省重点领域研发计划(2021B0101190004)、国家自然科学基金(62172157,62202149)、湖南省科技项目(2023GK2002、2021RC3062)、广东省自然科学基金(2023A1515012915)、深圳市基础研究面上项目(JCYJ20210324135409026)、之江实验室开放课题(2022RC0AB03)资助.**肖国庆**,博士,副教授,中国计算机学会(CCF)高级会员,主要研究方向为高性能计算、并行与分布式处理、人工智能与大数据计算.E-mail:xiaoguoqing@hnu.edu.cn.**李雪琪**,博士研究生,主要研究方向为大规模图神经网络和推荐系统.**陈玥丹**(通信作者),博士,副教授,中国计算机学会(CCF)会员,主要研究方向为高性能计算、并行与分布式处理.E-mail:chenyuedan@hnu.edu.cn.**唐 卓**,博士,教授,主要研究领域为超级计算与云计算、高性能计算与人工智能融合计算.**姜文君**,博士,教授,主要研究领域为社交网络分析、推荐系统.**李肯立**,博士,教授,中国计算机学会(CCF)会士,主要研究领域为高性能计算系统软件与应用.

methods used to deal with these challenges. In addition, we compare our survey with existing surveys on GNNs. The major difference is that our survey focuses specifically on GNN models in large-scale applications. We summarize and analyze related works on GNN algorithms and framework optimization with a focus on scalability. In the second section, we provide a brief overview of the message passing mechanism and classify GNN models into four categories: Graph Convolutional Networks, Graph Attention Networks, Graph Recurrent Neural Networks, and Graph Autoencoder. For each category, we introduce the major network design, including propagation and aggregation strategies, and analyze the corresponding challenges of processing large-scale data. Furthermore, we provide a summary of the challenges faced by GNN models in large-scale applications, in terms of full-batch and mini-batch training modes. Thirdly, we classify and analyze GNN algorithms for large-scale data. We focus on sampling-based GNNs at different granularities, which use node-, layer-, and subgraph-based sampling strategies to optimize the mini-batch training of GNNs. Specifically, node-based sampling strategies usually select a fixed number of neighbors for each node, layer-based sampling methods operate at each GNN layer, and subgraph-based sampling approaches attempt to find dense subgraphs as mini batches. We provide a summary of each type of sampling strategy, including its key ideas, related works, and a discussion of its advantages and disadvantages. In the fourth section of this survey, we introduce mainstream programming frameworks for GNN models and related optimization techniques for framework acceleration. We briefly introduce mainstream programming frameworks one by one, such as DGL, PyG, Graph-Learn, and also summarize their characteristics. We divide these optimization strategies into five categories: data partition, task scheduling, parallel execution, memory management, and other methods. Finally, we summarize this survey. We also provide prospects for future work in optimizing GNN models and accelerating frameworks for large-scale data, such as reducing redundant computation, algorithm and framework co-optimization, graph-aware optimizations, support for complex graphs, flexible scheduling based on hardware features, optimizations on distributed platforms, framework and hardware co-optimization and minimizing node representation dimensions.

Keywords　graph neural network; large-scale data; algorithm optimization; framework acceleration

1　引　言

图结构能够有效表征自然界和社会生活中广泛存在的复杂关系，如基因结构[1-2]、通信网络[3]、交通路线[4]、社交网络[5-6]等.针对相互关联的图数据，图计算能够有效挖掘其结构信息，但是不具备对其节点特征(比如，社交网络中用户兴趣、年龄等附加信息)的学习能力.另一方面，神经网络模型通过海量参数训练能够灵活表示数据的复杂特征，在图片、视频、文本等基于欧氏空间的数据上表现出优越的性能[7-9]，但其无法直接应用于非欧氏空间的图数据.综合图计算和神经网络的优势，图神经网络(Graph Neural Network，GNN)[10-11]凭借其在处理非欧氏空间数据及其复杂特征方面的优越性受到了大量的关注，并被广泛应用于网络链接预测[12]、推荐系统[13-14]、交通道路分析[15]等场景中.

实际应用场景中通常需要处理海量数据.2019年，阿里巴巴研究团队在其工作[16]中提到常见的电商平台往往存在百亿级节点、千亿级边，其存储开销超过10 TB.在面向大规模数据的图神经网络中，图数据的不规则性、特征复杂性以及训练过程中样本间的依赖性对模型计算效率、内存管理以及分布式系统中的通信开销造成了巨大的压力.下面我们将图神经网络模型在大规模数据应用中面临的挑战按照来源分为图数据结构、图神经网络模型、数据规模和硬件平台四类，分别介绍.

(1)图数据结构.图结构数据的不规则性、稀疏性、动态性、节点邻居数量呈幂律分布以及样本间的相互依赖，对高效的访存造成了一定的压力，尤其对面向大规模数据场景的分布式计算系统提出了更大的挑战.

(2)图神经网络模型. 节点的高纬表示是图神经网络模型区别于传统图算法的典型特征,在增强了模型表示能力的同时,也造成了更大的计算和内存开销,尤其是在大规模数据的应用中面临更大的挑战. 其迭代更新的机制设计使得深层的图神经网络模型面临邻居节点爆炸的问题.

(3)数据规模. 典型的图神经网络模型采用整批训练的模式,这在大规模数据的应用中存在内存限制的问题. 在基于分批训练模式的图神经网络模型中,数据大规模性增大了数据划分和迭代更新的难度.

(4)硬件结构. 图神经网络模型在图数据结构和复杂特征方面的建模需求使得模型既需要灵活的不规则数据读取,又需要高效的密集计算. 目前 CPU 在灵活读取方面表现较好,GPU 支持高效的密集计算,但二者都不能同时满足这两点需求. 图神经网络的需求和硬件结构的不匹配增加了大规模图神经网络模型加速的难度.

为应对和缓解上述困难和挑战,研究者通过优化硬件结构、编程框架、算法模型以及应用模型(如图 1 所示),来提高图神经网络模型的可拓展性、加速模型运行过程以及减小内存开销[17-18].

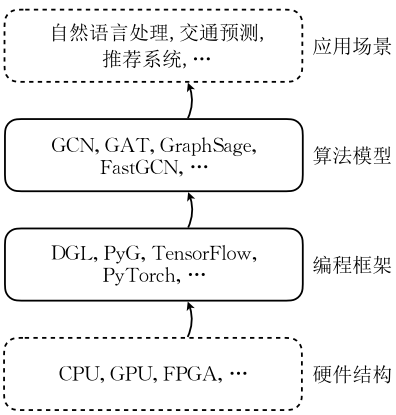

图 1 GNN 整体框架图

在应用模型方面,针对自然语言处理、交通道路预测、推荐系统等不同的应用场景提出具体的处理策略[4,13,19],以提高图神经网络模型在具体任务上的处理效率.

在算法模型方面,典型的图神经网络模型采用整批训练的方式,要求将完整的图数据载入内存,在面向大规模数据训练(尤其是基于 GPU 进行训练)时往往面临内存不足的问题. 针对这一问题,一些研究工作提出通过采样算法实现图神经网络模型的分批训练,比如 GraphSage[20]、FastGCN[21]、Cluster-GCN[22]等.

在编程框架方面,图神经网络迭代更新的表示机制导致训练样本之间相互依赖,使得 TensorFlow、Pytorch 等典型的神经网络框架无法有效地实现模型分批训练及其高效运行. 计算过程中的特征复杂性对原有图计算框架提出了更大的内存和计算需求. 结合传统神经网络和图计算框架的特点,Deep Graph Library(DGL)[23]、PyTorch Geometric(PyG)[24]等面向图神经网络的编程框架被提出来缓解这一问题.

在硬件结构方面,一些研究者结合 CPU、GPU、FPGA 等硬件的结构特征,在计算、访存等方面提出了对应的优化策略[25-27]. 或者针对图神经网络的特征设计了专用的硬件加速结构,比如 HyGCN[28].

基于以上大量的研究工作,目前已有综述从应用模型、算法模型、编程框架以及硬件结构四个方面总结和分析图神经网络相关的进展,表 1 列出了本文相关的 5 篇综述.

综述[29]总结了图神经网络在算法和应用方面的主要进展,将 GNN 算法分为循环 GNN、图卷积神经网络(Graph Convolutional Network,GCN)、图自编码器和时空 GNN 四类进行介绍,并总结了开源的数据集、已有方法的实现和应用场景. 综述[30]将图神经网络模型的设计过程总结为消息传播模式、采样方法和池化操作三个子模块的设计,并总结了不同的图神经网络模型和应用场景. 除了分类介绍 GNN 算法模型和应用场景,综述[31]还总结了已有的典型框架:消息传播网络(Message Passing Neural Network,MPNN)、非局部神经网络(Non-Local Neural Network,NLNN)、图网络(Graph Network,GN)和混合网络模型(Mixture Model Networks,MoNet). 综述[32]主要从算法模型和应用场景两方面总结了图卷积神经网络(GCN)的提出、发展和应用. 综述[33]主要总结了 GNN 在算法、软硬件加速器方面的进展,首先从计算过程的角度介绍了 GNN 的基础操作和算法分类,然后从软件(编程框架)和硬件两个方面总结了 GNN 加速器相关的工作,最后展望了今后加速器设计可能的方向:软硬件结合、图感知和以通信为中心. 综述[34]总结了现有图神经网络编程框架的设计和实现方案,分类分析了其中的优化技术,并对开源的图神线

表 1　图神经网络相关综述

文献	主要内容	侧重方面			
		应用	算法	框架	硬件
A Comprehensive Survey on Graph Neural Networks[29]	• 算法分类 • 应用：数据集、开源代码、应用场景	✓	✓		
Graph neural networks：A review of methods and applications[30]	• 图神经网络概述 • 算法模型分类总结 • 应用场景	✓	✓		
图神经网络综述[31]	• 算法分类 • 通用框架 • 应用场景	✓	✓		
图卷积神经网络综述[32]	• GCN 简介 • GCN 算法分类 • 应用场景	✓	✓		
Computing Graph Neural Networks：A Survey from Algorithms to Accelerators[33]	• 算法分类 • 加速器(软件、硬件)		✓	✓	✓
大规模图神经网络系统综述[34]	• 图神经网络概述 • 图神经网络编程框架 • 主要优化技术 • 实验分析			✓	
图神经网络加速结构综述[35]	• GNN 简介 • 硬件结构设计 • 关键技术：计算，片内、片外访存				✓
A Comprehensive Survey on Distributed Training of Graph Neural Networks[36]	• 分布式 GNN 训练模式介绍 • 软件框架和硬件平台设计 • 与分布式 DNN 训练的比较			✓	✓
Distributed Graph Neural Network Training：A Survey[37]	• 分布式 GNN 相关优化技术介绍 • 软件框架总结		✓	✓	
Parallel and Distributed Graph Neural Networks：An In-Depth Concurrency Analysis[38]	• GNN 简介 • GNN 的并行优化技术总结 • 相关的软件框架和硬件平台介绍	✓	✓	✓	✓
本文	• GNN 简介 • GNN 采样算法 • GNN 框架技术：框架及相关技术		✓	✓	

网络编程框架进行了实验评估. 综述[35]首先概述了 GNN 基础知识、典型算法、应用场景和主流编程框架，然后介绍了 GNN 加速结构的整体设计，并从 GNN 加速面临的挑战出发详细介绍了计算、片内访存、片外访存方面的关键技术.

总的来说，综述[29-32]侧重于总结和分析采用整图(full-batch)训练模式的图神经网络模型及其相关应用场景. 然而，当图中的节点或边的数量达到百万甚至十亿级时，训练过程往往会超出单块 GPU 的内存限制. 针对这一问题，一些方法的提出促进了图神经网络模型从全图训练方式到分批(mini-batch)训练方式的转变。其中，采样算法为图神经网络模型的分批训练提供支持，为其在大规模数据中的应用奠定了基础. 面向图神经网络的编程框架结合了深度学习框架和图结构的具体特征，提高了存储利用率和计算效率，促进了图神经网络模型在大规模数据中的应用. 相关综述[33-34]主要总结了图神经网络编程框架方面的进展. 综述[36-38]主要针对分布式平台，总结和分析了分布式 GNN 在算法模型、软件框架和硬件平台等方面的相关进展.

与现有综述相比，本文主要针对大规模图神经网络，从算法模型和框架优化两个方面对现有研究进行了调研、总结和分析. 本文首先对 GNN 的基础知识和典型算法进行介绍和总结；总结分析了基于不同粒度采样策略的图神经网络模型；总结分析了主流的加速框架及其相关技术. 为后续图神经网络模型在大规模数据数据应用中框架-算法的协同优化提供更多的思路.

本文的内容安排如图 2 所示. 第 2 节简要介绍图神经网络，包括消息传递机制、常见的图神经网络模型及其在针对大规模数据训练时面临的困难和挑战；第 3 节分类总结和分析图神经网络模型针对大规模数据的优化工作，主要包含对不同粒度的采样算法的介绍；第 4 节介绍和总结图神经网络编程框架加速方面的进展，包括对主流框架的介绍和相关加速技术的分类总结；第 5 节对全文进行总结并对面向大规模数据的图神经网络在算法模型和编程框架方面的未来研究进行展望.

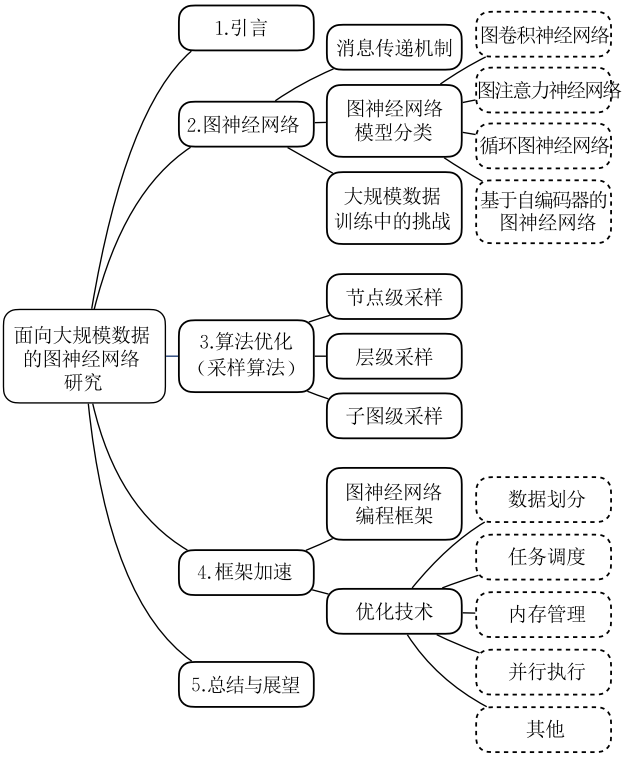

图 2 本文内容组织

2 图神经网络

图神经网络(Graph Neural Network,GNN)是面向图结构数据的神经网络模型[10-11],主要用于对图数据中的节点进行向量表示及其相关任务[39-40],融合了图计算和神经网络模型的优势,能够在捕捉图结构的同时抽象出节点包含的复杂特征[41].其中,图计算模型能够很好地捕捉图的拓扑结构特征,但无法处理节点的高维特征.典型神经网络模型通常用于处理欧氏空间数据,比如,卷积神经网络适用于处理网格类数据[42],循环神经网络在捕捉序列信息方面表现较好[43].综上,针对非欧氏空间的复杂图数据,图结构本身不规则并且节点包含复杂特征,其建模过程需要一种新的处理机制.目前比较受欢迎的消息传播模式通过获取高阶邻居信息来提升节点的表达能力,主要包括邻居聚合和节点更新两个步骤[44].

本节将从消息传递机制出发.首先简要介绍基于此的图神经网络模型中的两个主要操作,聚合和更新操作.然后分类介绍常见的图神经网络模型:图卷积神经网络、图注意力网络、循环图神经网络以及基于自编码器的图神经网络.并针对每一类图神经网络模型,分析其在大规模数据训练过程中存在的挑战.最后对图神经网络模型在面向大规模数据训练过程中存在的挑战进行总结.相关的符号及含义如表 2 所示.

表 2 符号定义

符号	表示含义
G	图
$\mathcal{V}$	节点集合,包含$\|\mathcal{V}\|$个节点
$\mathcal{N}_u$	节点 u 的邻居节点集合
$\boldsymbol{A}\in\mathbb{R}^{\|\mathcal{V}\|\times\|\mathcal{V}\|}$	邻接矩阵
$\boldsymbol{D}\in\mathbb{R}^{\|\mathcal{V}\|\times\|\mathcal{V}\|}$	度矩阵
$\boldsymbol{X}\in\mathbb{R}^{\|\mathcal{V}\|\times d_{ini}}$	所有节点的初始化特征
$\boldsymbol{E}\in\mathbb{R}^{\|\mathcal{V}\|\times d}$	所有节点的向量表示
$\boldsymbol{e}_u$	节点 u 的向量表示

2.1 消息传递机制

基于神经网络的消息传递机制描述了节点特征在网络中进行传播的过程,传播结果最终会通过神经网络操作迭代地更新在节点表示中[44].其中,第 k 次的消息传递过程如下:

$$\boldsymbol{h}_u^{(k)}=Aggregate^{(k)}(\{\boldsymbol{e}_v^{(k-1)},\forall v\in\mathcal{N}_u\}) \tag{1}$$

$$\boldsymbol{e}_u^{(k)}=Update^{(k)}(\boldsymbol{h}_u^{(k)},\boldsymbol{e}_u^{(k-1)}) \tag{2}$$

第 k 轮的聚合操作 $Aggregate^{(k)}$ 函数通过聚合节点 u 的邻居集合$\mathcal{N}_u$生成中间结果 $\boldsymbol{h}_u^{(k)}$.更新操作 $Update^{(k)}$ 函数通过结合 $\boldsymbol{h}_u^{(k)}$ 和节点 u 前一轮的表示 $\boldsymbol{e}_u^{(k-1)}$ 更新在 k 次迭代后的表示 $\boldsymbol{e}_u^{(k)}$.其中,聚合操作 $Aggregate$ 和更新操作 $Update$ 可以是任意可微函数,并且不同层的函数可以不同.根据是否考虑邻居节点间的先后顺序,已有的聚合操作大致分为基于集合的聚合和基于序列的聚合操作两类.

初始化函数 f_x 根据节点特征 $\boldsymbol{x}_u$ 生成其初始表示 $\boldsymbol{e}_u^{(0)}$,通过 K 轮消息传递过程分别得到对应的表示 $\boldsymbol{e}_u^{(k)}(k=1,2,\cdots,K)$,最终表示函数 f_e 基于此生成最终的节点表示 $\boldsymbol{e}_u$.

$$\boldsymbol{e}_u^{(0)}=f_x(\boldsymbol{x}_u) \tag{3}$$

$$\boldsymbol{e}_u^{(k)}=Update^{(k)}(Aggregate^{(k)}(\{\boldsymbol{e}_{\mathcal{N}_u}^{(k-1)}\}),\boldsymbol{e}_u^{(k-1)}) \tag{4}$$

$$\boldsymbol{e}_u=f_e(\{\boldsymbol{e}_u^{(k)},k=1,2,\cdots,K\}) \tag{5}$$

通过迭代的聚合和更新过程(如式(4)所示),每个节点将会包含其高阶邻居的特征.比如,$\boldsymbol{e}_u^{(1)}$ 通过聚合邻居节点 $\boldsymbol{e}_v^{(0)}(v\in\mathcal{N}_u)$包含了一阶邻居的特征,$\boldsymbol{e}_u^{(2)}$ 通过聚合邻居节点 $\boldsymbol{e}_v^{(1)}(v\in\mathcal{N}_u)$包含了一阶和二阶邻居(邻居的邻居)的特征.类似地,通过 K 次迭代聚合和更新,$\boldsymbol{e}_u^{(K)}$ 将包含一阶,二阶,…,K 阶邻居的特征,包含了图结构信息.另一方面,$\boldsymbol{e}_u^{(0)}$ 根据其本身特征生成.总的来说,图神经网络表示模型能够捕捉图结构信息和建模复杂的节点特征.

2.2 常见模型

本节将介绍常见的四种图神经网络模型，图卷积神经网络、图注意力网络、循环图神经网络以及基于自编码器的图神经网络，并具体分析其在大规模数据训练中存在的挑战.

2.2.1 图卷积神经网络

图卷积神经网络(Graph Convolutional Network，GCN)是目前最常见的图神经网络模型[45]，通过卷积操作来实现邻居节点聚合. 根据卷积操作的定义方式，目前 GCN 模型主要分为基于谱域和基于空间域两类，综述[32]提供了详细的介绍. 其中，基于谱域的方法基于图信号分析和图谱论相关的工作，基于空间域的方法主要关注邻居节点的直接聚合方式. 下文将对二者进行简单介绍.

基于谱域的方法从图的层面来定义卷积操作[45].

$$\boldsymbol{E}^{(k+1)}=f(\widetilde{\boldsymbol{D}}^{-\frac{1}{2}}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{D}}^{-\frac{1}{2}}\boldsymbol{E}^{(k)}\boldsymbol{W}^{(k+1)}) \tag{6}$$

其中，$\boldsymbol{E}^{(k)}\in\mathbb{R}^{|\mathcal{V}|\times d}$代表包含 k 阶邻居特征的节点表示矩阵，$|\mathcal{V}|$ 表示节点个数，d 表示嵌入维度. $\widetilde{\boldsymbol{A}}=\boldsymbol{A}+\boldsymbol{I}$，$\boldsymbol{A}$ 表示邻接矩阵，$\boldsymbol{I}$ 是单位矩阵. $\widetilde{\boldsymbol{D}}$ 是对角矩阵，用于归一化处理，$\widetilde{\boldsymbol{D}}_{i,i}=\sum_j\widetilde{\boldsymbol{A}}_{i,j}$. $f(\cdot)$代表非线性激活函数，$\boldsymbol{W}^{(k+1)}$ 表示特征转化函数. 基于此，一些方法将最终的迭代结果作为节点表示，如式(7)所示. 另一方法则综合多层迭代结果来进行节点表示，如式(8)所示，$f_{comb}(\cdot)$表示结合函数，常见形式有均值、加权均值、向量拼接等.

$$\boldsymbol{E}=\boldsymbol{E}^{(K)} \tag{7}$$

$$\boldsymbol{E}=f_{comb}(\{\boldsymbol{E}^{(k)},k=0,1,\cdots,K\}) \tag{8}$$

而基于空间域的方法则从节点的角度来考虑卷积操作，一个节点的状态是通过聚合其邻居节点的特征来进行更新的，具体的聚合和更新操作如下：

$$\boldsymbol{h}_u^{(k+1)}=\sum_{v\in\mathcal{N}_u}\frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_v|}}\boldsymbol{e}_v^{(k)}\boldsymbol{W}^{(k+1)} \tag{9}$$

$$\boldsymbol{e}_u^{(k+1)}=f\left(\boldsymbol{h}_u^{(k+1)}+\frac{1}{|\mathcal{N}_u|}\boldsymbol{e}_u^{(k)}\boldsymbol{W}^{(k+1)}\right) \tag{10}$$

其中，$|\mathcal{N}_u|$表示节点 u 的邻居数量，$\frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_v|}}$进行归一化操作，避免数值不稳定和梯度爆炸的问题. 其矩阵形式如式(6)所示，和基于谱域的卷积操作一致，图卷积神经网络架起了基于谱域和基于空间域的卷积操作的桥梁[46].

在大规模数据训练中存在的挑战. 图卷积神经网络通常采样整批训练的方式(如式(6)所示)，训练过程要求将整个邻接矩阵载入内存，这在面向大规模数据的训练中往往存在内存不足的限制. 针对此问题，一些工作提出了分批训练的方式，对节点进行分批，在每一批次内，通过式(9)和(10)迭代更新每一个节点的表示. 设每一批次包含 B 个节点，每一个节点平均有 d 个邻居，卷积神经网络包含 k 层，则每一批数据的计算复杂度是 $O(Bd^k)$. 每个节点占用 m 字节的内存，则单个批次需要消耗 mBd^k 字节的内存资源. 随着图神经网络层数目 k 的增加，计算和内存资源的消耗呈指数级增长. 总的来说，在面向大规模数据的训练过程中，整批训练的图卷积神经网络模型存在内存不足的限制，分批训练的图卷积神经网络模型面临邻居爆炸的问题.

2.2.2 图注意力网络

注意力机制通过引入可训练的权重参数[47]，来区分元素对目标任务的不同贡献，在自然语言处理、图像识别等领域有广泛的应用.

在图结构中，节点和节点之间的关联性也存在差异，基于此，Velickovic 等人[48]提出了图注意力网络(Graph Attention Network，GAT). GAT 的基本形式表示如下：

$$\boldsymbol{h}_u^{(k)}=\boldsymbol{W}^{(k)}\boldsymbol{e}_u^{(k-1)} \tag{11}$$

$$\hat{\boldsymbol{\alpha}}_{u,v}^{(k)}=f^1(\boldsymbol{a}^{(k)\mathrm{T}}[\boldsymbol{h}_u^{(k)}\,\|\,\boldsymbol{h}_v^{(k)}]) \tag{12}$$

$$\boldsymbol{\alpha}_{u,v}^{(k)}=\frac{\exp(\hat{\boldsymbol{\alpha}}_{u,v}^{(k)})}{\sum_{i\in\mathcal{N}_u}\exp(\hat{\boldsymbol{\alpha}}_{u,i}^{(k)})} \tag{13}$$

$$\boldsymbol{e}_u^{(k)}=\sigma\Big(\sum_{v\in\mathcal{N}_u}\boldsymbol{\alpha}_{u,v}^{(k)}\boldsymbol{e}_v^{(k-1)}\Big) \tag{14}$$

其中，$\boldsymbol{\alpha}_{u,v}^{(k)}$是注意力权重，表示在第 k 层聚合过程中邻居节点 v 对表示节点 u 的重要性，$\boldsymbol{a}^{(k)}\in\mathbb{R}^{2\times d_k}$ 表示权重向量，d_k 是经过线性转化之后的特征维度，$\boldsymbol{W}^{(k)}\in\mathbb{R}^{d_{k-1}\times d_k}$ 是特征转化矩阵. f_1 是非线性激活函数，GAT 将 f_1 具体化为 LeakyReLU. 此外，GAT 也提供了多头注意力机制在图神经网络中的实现，即针对同一个节点的同一个邻居有不同的权重.

在大规模数据训练中存在的挑战. 和图卷积神经网络相比，图注意力网络的不同点在于基于注意力权重进行加权聚合(如式(14)所示)，而典型图卷积神经网络采用标准化求和进行节点聚合(如式(9)所示). 在大规模数据训练中，图卷积神经网络存在内存不足和邻居爆炸问题，在图注意力网络模型中依旧存在，并且注意力权重的计算和存储(式(12)和(13))需要消耗更多的计算和内存资源.

2.2.3 循环图神经网络

信息序列是推荐系统、交通预测、音频处理等应

用中常见的形式之一，比如文本序列、用户历史浏览记录、音视频等.在序列中，一个元素的作用会随着时间的增长而减弱.为了准确建模序列信息、捕捉其长短期依赖特征，研究者们提出了循环神经网络模型，其常见形式是长短期记忆网络(Long Short-Term Memory Network，LSTM)[43]，门控循环单元(Gate Recurrent Units，GRU)[49].

Li 等人基于 GRU 提出了针对图结构的循环图神经网络模型 GGNN(Gated Graph Neural Network)[50]，主要针对以输出状态序列为目标的任务，常见的图神经网络模型以输出单个状态表示为目标.循环图神经网络传播过程表示如下：

$$\boldsymbol{a}_v^{(k)}=\boldsymbol{A}_v^{(k)\mathrm{T}}[\boldsymbol{e}_1^{(k-1)},\boldsymbol{e}_2^{(k-1)},\cdots,\boldsymbol{e}_N^{(k-1)}]+\boldsymbol{b}^{(k)} \tag{15}$$

$$\boldsymbol{z}_v^{(k)}=\sigma(\boldsymbol{W}^z\boldsymbol{a}_v^{(k)}+\boldsymbol{W}^z\boldsymbol{e}_v^{(k-1)}) \tag{16}$$

$$\boldsymbol{r}_v^{(k)}=\sigma(\boldsymbol{W}^r\boldsymbol{a}_v^{(k)}+\boldsymbol{U}^r\boldsymbol{e}_v^{(k-1)}) \tag{17}$$

$$\tilde{\boldsymbol{e}}_v^{(k)}=\tanh(\boldsymbol{W}\boldsymbol{a}_v^{(k)}+\boldsymbol{U}(\boldsymbol{r}_v^{(k)}\odot\boldsymbol{e}_v^{(k-1)})) \tag{18}$$

$$\boldsymbol{e}_v^{(k)}=(1-\boldsymbol{z}_v^{(k)})\odot\boldsymbol{e}_v^{(k-1)}+\boldsymbol{z}_v^{(k)}\odot\tilde{\boldsymbol{e}}_v^{(k)} \tag{19}$$

首先对节点 v 的邻居节点 $(\boldsymbol{e}_1^{(k-1)},\boldsymbol{e}_2^{(k-1)},\cdots,\boldsymbol{e}_N^{(k-1)})$ 进行聚合生成中间表示 $\boldsymbol{a}_v^{(k)}$，其中 $\boldsymbol{A}_v^{(k)\mathrm{T}}$ 表示在第 k 个时间步，和节点 v 相关的子邻接矩阵，$\boldsymbol{b}^{(k)}$ 是偏置因子.$\boldsymbol{z}_v^{(k)}$ 代表第 k 个时间步的更新门，决定了应当保留的信息特征，$\boldsymbol{W}^z$ 和 $\boldsymbol{W}^z$ 是其相关的系数矩阵.$\boldsymbol{r}_v^{(k)}$ 代表重置门，决定了应当丢弃的信息特征，$\boldsymbol{W}^r$ 和 $\boldsymbol{U}^r$ 是其相关的系数矩阵.$\tilde{\boldsymbol{e}}_v^{(k)}$ 是经过了重置门 $\boldsymbol{r}_v^{(k)}$ 过滤之后的中间表示，$\odot$ 是哈达玛积(Hardamard product).$\boldsymbol{e}_v^{(k)}$ 是结合了更新门 $\boldsymbol{z}_v^{(k)}$ 生成的特征后的第 k 步的最终表示.

式(15)到(19)描述了针对单个节点的更新过程，设 $\boldsymbol{A}_G^{(k)}$ 为图 G 在第 k 个时间步的邻接矩阵，循环图神经网络模型的矩阵形式如下：

$$\boldsymbol{A}^{(k)}=\boldsymbol{A}_G^{(k)\mathrm{T}}\boldsymbol{E}^{(k-1)}+B \tag{20}$$

$$\boldsymbol{Z}^{(k)}=\sigma(\boldsymbol{W}^z\boldsymbol{A}^{(k)}+\boldsymbol{W}^z\boldsymbol{E}^{(k-1)}) \tag{21}$$

$$\boldsymbol{R}^{(k)}=\sigma(\boldsymbol{W}^r\boldsymbol{A}^{(k)}+\boldsymbol{U}^r\boldsymbol{E}^{(k-1)}) \tag{22}$$

$$\widetilde{\boldsymbol{E}}^{(k)}=\tanh(\boldsymbol{W}\boldsymbol{A}^{(k)}+\boldsymbol{U}(\boldsymbol{R}^{(k)}\odot\boldsymbol{E}^{(k-1)})) \tag{23}$$

$$\boldsymbol{E}^{(k)}=(1-\boldsymbol{Z}^{(k)})\odot\boldsymbol{E}^{(k-1)}+\boldsymbol{Z}^{(k)}\odot\widetilde{\boldsymbol{E}}^{(k)} \tag{24}$$

图卷积神经网络模型和图注意力模型以静态图作为输入，通过多层神经网络迭代地捕捉图结构特征.而循环图神经网络的输入图随着时间步演化，借助遗忘门、更新门等结构对图结构的演化特征进行建模.

在大规模数据训练中存在的挑战. 和图卷积神经网络模型相比，整批训练的循环图神经网络同样要求将整个邻接矩阵载入内存，并且需要更大的内存，设一共 K 个时间步，每个邻接矩阵占用 m 字节的内存，则图数据一共占用 mK 字节的内存(而在图卷积神经网络中，单一状态的图数据只占用 m 字节的内存).训练过程涉及 $\boldsymbol{W}^z$、$\boldsymbol{W}^z$、$\boldsymbol{W}^r$、$\boldsymbol{U}^r$、$\boldsymbol{W}$、$\boldsymbol{U}$ 等大量的参数，也需要大量的内存.在针对大规模数据训练时主要面临内存方面的挑战.在分批训练模式中，如式(15)所示，其演化过程涉及直接邻居的节点表示，图数据的不规则性在一定程度上增加了冗余计算量.

2.2.4 基于自编码器的图神经网络

自动编码器能够通过无监督学习的方式高效学习节点表示[51-52]，由编码器和解码器两个部分构成.编码器通过多层神经网络结构将输入空间的特征 $\boldsymbol{X}$ 映射到潜在空间 $\boldsymbol{Z}$，解码器采用和编码器对称的神经网络结构，将 $\boldsymbol{Z}$ 解码到输入空间，记作 $\hat{\boldsymbol{X}}$.自编码器通过减小重构损失(即 $\boldsymbol{X}$ 和 $\hat{\boldsymbol{X}}$ 之间的差异性)来优化网络参数.

Wang 等人将自动编码器应用到图结构数据中，提出了 SDNE(Structural Deep Network Embedding)模型[53].SDNE 借助多层神经网络构成的编码器将节点 u 的输入特征 $\boldsymbol{x}_u$ 映射到潜在空间得到其嵌入表示 $\boldsymbol{z}_u$，再通过解码器重构节点特征 $\hat{\boldsymbol{x}}$，其具体过程如下：

$$\boldsymbol{y}_u^{(1)}=\sigma(\boldsymbol{W}^{(1)}\boldsymbol{x}_u+\boldsymbol{b}^{(1)}) \tag{25}$$

$$\boldsymbol{y}_u^{(k)}=\sigma(\boldsymbol{W}^{(k)}\boldsymbol{y}_u^{(k-1)}+\boldsymbol{b}^{(k)}) \tag{26}$$

$$\boldsymbol{z}_u=\boldsymbol{y}_u^{(K)} \tag{27}$$

$$\hat{\boldsymbol{y}}_u^{(k)}=\sigma(\hat{\boldsymbol{W}}^{(k)}\boldsymbol{y}_u^{(k+1)}+\hat{\boldsymbol{b}}^{(k)}) \tag{28}$$

$$\hat{\boldsymbol{x}}=\sigma(\hat{\boldsymbol{W}}^{(1)}\hat{\boldsymbol{y}}_u^{(1)}+\hat{\boldsymbol{b}}^{(1)}) \tag{29}$$

和典型自动编码器模型相同，SDNE 需要减小节点的重构损失 $L_{2\mathrm{nd}}$.此外，还考虑节点间的相似性 $s_{u,v}$(如果节点 u 和 v 有关联，则 $s_{u,v}$ 表示二者的相似性；如果节点 u 和 v 没有关联，则 $s_{u,v}$ 等于 0)，借助 $L_{1\mathrm{st}}$ 来减小相似节点在潜在空间的距离.L_{reg} 是正则化损失.

$$L_{2\mathrm{nd}}=\sum_{u=1}^{n}\|\hat{\boldsymbol{x}}_u-\boldsymbol{x}_u\|_2^2 \tag{30}$$

$$L_{1\mathrm{st}}=\sum_{u,v=1}^{n}s_{u,v}\|\boldsymbol{y}_u^{(k)}-\boldsymbol{y}_v^{(k)}\|_2^2 \tag{31}$$

$$L_{\mathrm{reg}}=\frac{1}{2}\sum_{k=1}^{K}(\|\boldsymbol{W}^{(k)}\|_F^2+\|\hat{\boldsymbol{W}}^{(k)}\|_F^2) \tag{32}$$

$$L=L_{2\mathrm{nd}}+\alpha L_{1\mathrm{st}}+\beta L_{\mathrm{reg}} \tag{33}$$

在大规模数据训练中存在的挑战. 和应用广泛

的图卷积神经网络相比，基于自动编码器的图神经网络模型不需要迭代地聚合邻居节点进行特征学习，无法捕捉节点间的高阶关联. 借助损失函数 L_{1st} (式(31))捕捉节点间的直接关联性. 在针对大规模数据的训练中，损失函数 L_{1st} 要求获得所有节点对的向量表示，受到内存限制自动编码器无法一次性生成全部的节点表示，需要分批训练获得. 大规模数据的分批训练使得不同批节点间的比较会产生大量的冗余计算. 即使采用负样本采样的形式来进行训练，图数据的不规则性也会对冗余计算的消减提出一定的挑战.

2.3　图神经网络在大规模数据应用中存在的挑战

2.2 节介绍了不同类型的图神经网络模型及其在大规模数据应用中存在的挑战，下面将这些模型层相关的挑战按照来源分为图神经网络模型、图数据结构以及数据规模三类，分别总结基于整批训练和分批训练模式的模型在这三个方面对应的挑战. 其简要概括如表 3 所示.

(1) 图神经网络模型相关的挑战. ① 高纬特征的运算需要消耗大量的计算和内存资源，尤其是在针对大规模图的分布式训练中节点的高维特征同步会造成较高的通信延迟；② 典型图神经网络模型采用整批训练的模式，要求将图的邻接矩阵载入内存，在大规模数据训练中存在内存不足的问题；③ 针对内存不足的问题，一些工作提出采用深度神经网络模型中分批训练的形式. 然而，在图神经网络中，迭代更新的表示机制使得节点的表示往往依赖于其邻居节点. 准确的表示需要迭代构建其高阶邻居，随着网络层数的增多会导致模型面临邻居节点爆炸的问题.

(2) 图结构数据相关的挑战. 在基于整批训练的模型方面，运算过程通常依赖于邻接矩阵，图结构数据的稀疏性在一定程度上增加了计算和内存开销. 在基于分批训练的模型方面，图结构的不规则性增加了数据划分的难度，此外，忽略不同批节点间的关联往往会造成一定的信息损失，而建模其关联则会产生一定的冗余计算. 尤其是基于分布式系统的分批训练模型中，节点间的同步会需要一定的通信开销.

(3) 数据规模相关的挑战. 一方面，图数据的大规模性导致基于整批训练的模型存在内存限制的问题. 另一方面，大规模性也对基于分批训练的模型在数据划分以及网络模型运行时的计算和内存开销提出了挑战.

表 3　图神经网络在大规模数据应用中存在的挑战

方面	模型训练方式	
	整批训练	分批训练
图神经网络模型	* 高维表示 · 计算开销大 · 内存开销大	* 迭代更新机制 · 邻居节点爆炸
图数据结构	* 稀疏性 · 冗余计算 · 冗余内存	* 不规则性 · 数据划分难 · 特征损失 · 冗余计算 · 通信开销(分布式)
数据规模	* 大规模性 · 内存限制	* 大规模性 · 数据划分 · 模型运行

注：* 表示产生相关挑战的主要原因.

3　图神经网络采样算法

针对图神经网络在大规模数据训练中面临的挑战，一些研究工作在算法模型方面进行了对应的优化. 大部分工作集中在针对数据的优化方面，主要通过不同粒度的采样算法实现分批训练以应对数据大规模性在计算效率和内存开销方面带来的挑战. 根据采样粒度，已有的采样算法可以分为基于节点的采样算法、基于层的采样算法以及基于子图的采样算法. 下文将分别介绍和分析三类采样算法及其涉及的具体模型.

3.1　基于节点的采样算法

针对图神经网络模型分批训练中图数据的不规则性造成的无法参数共享的问题(参数共享指不同样本中的同一特征在目标任务中发挥相似的作用，则其相应参数保持一致. 神经网络模型基于这一机制，通过海量数据进行参数训练以提升其建模能力)，一些研究者提出针对每个节点对其邻居节点进行采样(即基于节点的采样算法)，通过采样操作将图数据规则化以适应参数共享，为大规模数据的分批训练奠定了基础. 下面将介绍和分析 4 个基于节点的采样算法，GraphSage、PinSage、VR-GCN 和 LGCL.

GraphSage. 大部分图神经网络模型需要训练集中包含所有节点，这种直推式学习模型无法拓展到新的数据，即训练集以外的节点. 针对此问题，Hamilton 等人[20]提出了一个更为通用的归纳式学习框架 GraphSage，基于节点采样进行表示学习. 已有模型通常以生成节点表示为目标，而 GraphSage 则是以优化模型参数为目标. 如图 3(b)所示，针对目标节点，首先在每一阶邻居节点中随机采样固定

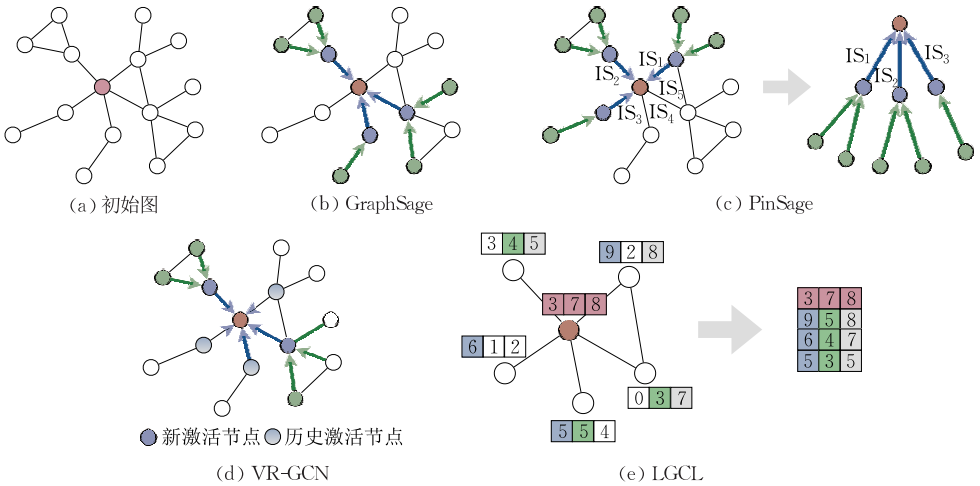

图 3 基于节点的采样算法

数目的节点,然后采用每一阶对应的聚合函数进行邻居节点特征聚合,并借助反向传播算法对聚合函数中的参数进行学习. 通过优化之后的模型能够用于表示新的数据.

GraphSage 借助随机节点采样算法,将不规则图结构数据规则化(每个节点有固定阶数的邻居,且每一阶邻居分别有固定数目的节点),从而实现不同数据间的参数共享.

PinSage. 结合随机游走和图卷积操作,Ying 等人提出了一个用于大规模推荐系统的采样算法 PinSage[54]并将其部署在图片社交网站 Pinterest. 为了提高图卷积神经网络模型在大规模数据上的可拓展性,PinSage 利用节点采样构建计算图来捕捉图结构特征,取代整批(full-batch)训练模型中的图拉普拉斯矩阵和特征矩阵之间的乘法运算. 提出了基于重要性的节点采样算法,如图 3(c)所示,利用随机游走策略评估节点的重要性,对每个节点选择对其最重要的 K 个节点作为其采样节点,并在其聚合过程中进行重要性加权.

VR-GCN. Chen 等人[55]提出现有采样算法虽然在一定程度上缓解了大规模图神经网络模型在参数共享方面存在的问题,但是却没有保证采样算法收敛性. 基于方差消减提出了新的采样算法 VR-GCN (Stochastic GCN with Variance Reduction),并从理论上证明了无论采样规模如何,VR-GCN 都能够达到局部最优性能. 如图 3(d)所示,针对每个节点,VR-GCN 仅采样两个节点,并利用历史激活节点(即当前批次训练中已完成计算的节点)来减小方差. 并通过实验表明相比于已有的采样算法,VR-GCN 能够显著减小估计梯度的偏置和方差. 相比于考虑所有邻居节点的情况,VR-GCN 仅考虑 2 个邻居节点也可以达到相同的性能,大大减小了模型训练的时间复杂度和内存开销.

LGCL. 卷积操作在欧式空间数据挖掘上表现良好,针对不规则的图数据,大部分图卷积神经网络模型将卷积操作拓展到图结构,Gao 等人[56]从另一个角度结合卷积操作和图神经网络,将图数据结构化以满足卷积操作的要求. 首先,利用 $g(\cdot)$根据当前层的节点表示 $\boldsymbol{E}_k \in \mathbb{R}^{|\mathcal{V}|\times d}$和邻接矩阵 $\boldsymbol{A} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$,采样 n 个最大节点和自身合并组成 $\hat{\boldsymbol{E}}_k \in \mathbb{R}^{|\mathcal{V}|\times(n+1)\times d}$,将图数据转化到欧氏空间. 然后再利用卷积操作 $c(\cdot)$进行特征学习,具体过程如下:

$$\hat{\boldsymbol{E}}^{(k)} = g(\boldsymbol{E}^{(k-1)}, \boldsymbol{A}, n) \tag{34}$$

$$\boldsymbol{E}^{(k)} = c(\hat{\boldsymbol{E}}^{(k)}) \tag{35}$$

此外,针对大规模数据训练中的内存限制和计算开销大的问题,LGCL 采用基于子图的训练方法,基于宽度优先算法通过逐层扩张的方法生成子图.

与 GraphSage 和 PinSage 从原节点集合中采样不同,如图 3(e)所示,LGCL 在每个特征维度选择最大的 n 个特征值组成 n 个采样节点. LGCL 通过节点特征重组将不规则的图结构数据转化到欧氏空间,便于利用已有的卷积神经网络(CNN)算法进行优化. 但是基于显著特征的重组方式在一定程度上破坏了节点特征的多样性,加剧了节点表示过度平滑的问题. 以图 3(e)为例,采样均值聚合的方式,通过一阶邻居的聚合更新之后,目标节点的第一个特征值更新为 5.75,通过多轮特征传播之后,每个节点的特征值都会趋于接近对应特征的最大值. 最终,所有节点的表示趋于相似,加剧图卷积神经网络的过度平滑的问题.

基于节点的采样算法总结. 针对图神经网络中直推式训练模型存在的局限性,GraphSage 首先提

出了基于节点的采样算法，通过随机的节点采样算法将不规则图结构数据规则化以适应归纳式任务. 如图 3(b)所示，GraphSage 随机采样一阶和二阶邻居，其中一阶采样 3 个节点、二阶采样 2 个节点. 不同于 GraphSage 的随机采样，PinSage 认为不同节点的贡献有所不同，提出了基于重要性的采样算法，并在节点聚合中进行重要性加权. GraphSage、PinSage 主要对采样策略进行了优化，而 VR-GCN 则更加关注采样算法的收敛性，通过减小梯度估计的偏置和方差来提高算法收敛性. 以上采样算法基于已有节点进行操作，LGCL 则是从特征粒度进行筛选重组为新的节点进行聚合.

3.2　基于层的采样算法

基于节点的采样算法为面向大规模数据的图神经网络模型的分批训练奠定了基础，但是在实际应用中采样节点的数量随着图网络层数的增加呈指数级增长，造成了邻居节点爆炸的问题. 针对这一问题，一些研究者提出了基于层的采样算法，针对图神经网络的每一层，采样确定数目的节点. 下面将介绍和分析 3 个基于层的采样算法，FastGCN、AS-GCN 和 LADIES.

FastGCN. 迭代式的邻居扩张在图神经网络的大规模数据训练中造成了巨大的时间和内存开销，针对这一挑战，Chen 等人[21]提出将图卷积操作转化为以概率分布为表示函数的积分形式(如图 4(a)所示)，并利用蒙特卡洛法采样来估计积分值，并将这一方法命名为 FastGCN. 作者假设图是无限大的，输入图是无限大图的一个子图，且每个节点满足独立同分布，$v_0, v_1, \cdots, v_{|\mathcal{V}|} \sim P$，则式(36)所示的卷积过程可以转化为式(37)所示的积分形式.

$$\boldsymbol{E}^{(k)} = \hat{\boldsymbol{A}}\sigma(\boldsymbol{E}^{(k-1)})\boldsymbol{W}^{(k)} \tag{36}$$

$$\boldsymbol{e}_u^{(k)} = \int \hat{\boldsymbol{A}}_{u,v}\sigma(\boldsymbol{e}_u^{(k-1)})\boldsymbol{W}^{(k)}\,dP(v) \tag{37}$$

其中，$\boldsymbol{e}_u^{(k)}$ 是节点 u 在第 k 层的表示，σ 表示非线性激活函数，$\boldsymbol{W}^{(k)}$ 是第 $k-1$ 到 k 层的特征转化函数. $\hat{\boldsymbol{A}}_{u,v}$ 表示节点 v 在节点 u 表示过程中的共享. FastGCN 利用蒙特卡洛法估计积分值，在第 k 层采样 t_k 个独立同分布样本 $v_{t_1}, v_{t_2}, \cdots, v_{t_k} \sim P$，来估算式(37)的值，记作 $\boldsymbol{e}_{t_{k+1}}^{(k)}(u)$，

$$\boldsymbol{e}_{t_{k+1}}^{(k)}(u) := \frac{1}{t_k}\sum_{j=1}^{t_k}\hat{\boldsymbol{A}}_{u,v_j}\sigma(\boldsymbol{e}_{t_k}^{(k-1)}(v_j^{(k)}))\boldsymbol{W}^{(k)} \tag{38}$$

FastGCN 通过层级采样避免了图卷积神经网络中迭代更新造成的邻居节点爆炸问题，基于式(38)相应地提出了基于采样的损失函数和梯度函数进行模型训练，并且提出重要性采样进一步优化模型性能.

AS-GCN. AS-GCN(Adaptive Sampling GCN)是一个自适应的层级采样算法[57]，通过在每一层采样固定数目的节点来避免 GCN 中邻居节点爆炸的问题. 按照从上到下的顺序构建 GCN 模型，基于上层采样结果对下层节点进行采样，使下层采样邻居节点被尽可能多的上层节点共享. 在采样算法训练过程中除了提高模型性能，加入显式的方差消减，其优化目标具体表示如下：

$$L = \frac{1}{n}\sum_{i=1}^{n}L_c(y_i, \bar{y}(\hat{\mu}_q(v_i)))\lambda Var_q(\hat{\mu}_q(v_i)) \tag{39}$$

其中，$L_c(y_i, \bar{y}(\hat{\mu}_q(v_i)))$ 表示节点 v_i 在分类任务上的损失函数，$Var_q(\hat{\mu}_q(v_i))$ 表示对应估计值的方差.

此外，如图 4(b)所示，AS-GCN 还通过连接跳

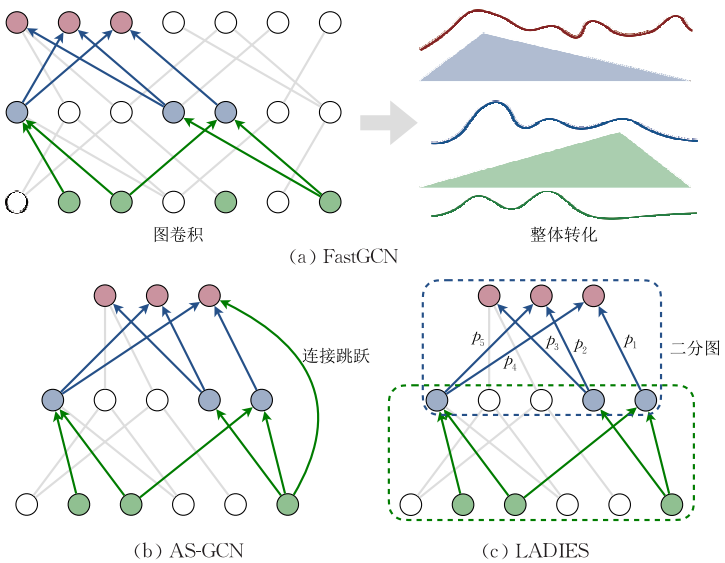

图 4　基于层的采样算法

跃(skip connections)来捕捉二阶相似性.典型的GCN模型在邻居节点聚合过程中只处理一阶邻居节点.为了更好地利用网络结构中高阶邻居的特征,AS-GCN利用连接跳跃策略获取二阶邻居的特征,即在 $l-1$ 和 $l+1$ 层之间添加连接用于节点聚合.能够在没有增加额外采样开销的前提下,传播高阶邻居的特征信息.

LADIES.基于节点的采样算法迭代式地针对每一个节点采样固定数目的邻居节点,随着图神经网络层数目的增加,采样节点数目呈指数级增长,造成大量的计算和内存开销.基于层的采样算法能够克服计算和内存开销过大的问题,每一层只需要计算固定数量的节点,但往往存在相邻层之间连接稀疏的问题.

针对以上基于节点和基于层的采样算法中存在的挑战,Zou等人[58]提出了新的采样算法LADIES(LAyer-Dependent ImportancE Sampling).如图4(c)所示,根据上层节点,LADIES对其邻居节点进行采样作为下层节点.具体来说,基于上层节点及其邻居节点构建二分图,并计算其重要性分数作为采样概率,依概率采样固定数目的邻居节点.再将采样生成的下层节点作为上层节点,迭代地构建整个计算图.

基于层的采样算法总结.FastGCN(如图4(a)所示)侧重于从宏观的角度,将图卷积过程转化为基于概率分布的积分形式,并通过层级采样来估计积分值,避免了邻居节点爆炸问题,但存在相邻层之间连接稀疏以及冗余节点的问题.AS-GCN(如图4(b)所示)通过在优化目标中加入显式的方差消减来保证模型的收敛性,并通过连接跳跃策略来捕捉二阶关联性.LADIES(如图4(c)所示)将相邻两层的节点构建为二分图,并基于此进行层级重要性采样.基于层的采样算法通过固定层级的采样节点数目,在一定程度上缓解了基于节点的采样算法存在的邻居节点爆炸问题,但是只考虑了相邻层之间的节点关联,在全局节点复用方面仍然存在一定的局限性.

3.3 基于子图的采样算法

在图神经网络模型训练中,整批训练的方式能够有效重用节点,避免冗余计算,但在处理大规模数据时往往存在内存不足的问题;分批训练的方式通过将数据分批放入内存能够避免这一问题,但不同批次中的同一节点往往需要重复计算.基于子图采样的图神经网络模型以综合二者的优点为目标,从大规模图数据中采样多个不同的子图以避免大规模图整批载入时内存不足的问题;针对每个子图采用整批训练的方式,以提高节点重用率和计算并行性.下文将介绍和分析4个比较常见的子图采样算法,ClusterGCN、RWT、GraphSAINT 和 SHADOW-GNN.

Cluster-GCN.为了提高GCN分批训练的计算效率,Chiang等人提出了节点表示利用次数(Embedding Utilization)的概念[22],如果节点 u 在第 i 层的表示 $\boldsymbol{e}_u^{(i)}$ 在第 $i+1$ 层的计算中被重用了 n 次,则节点表示利用次数为 n.并指出通过提高节点表示利用次数来提高训练效率,即数据分块内节点之间的边数.基于此动机,Chiang等人提出了子图采样算法Cluster-GCN[22].首先采用聚类感知的划分算法Metis[59]将节点划分为 c 块,$\mathcal{B}=\{b_1,b_2,\cdots,b_c\}$,并将邻接矩阵 $\boldsymbol{A}$ 转化为基于聚类划分的对角矩阵 $\overline{\boldsymbol{A}}$ 和 Δ,用 $\boldsymbol{A}_{ij}$ 表示分块 b_i 和 b_j 中包含的节点对应的邻接矩阵.

$$\boldsymbol{A}=\overline{\boldsymbol{A}}+\Delta=\begin{bmatrix}\boldsymbol{A}_{11} & \cdots & \boldsymbol{A}_{1c}\\ \vdots & \ddots & \vdots\\ \boldsymbol{A}_{c1} & \cdots & \boldsymbol{A}_{cc}\end{bmatrix} \tag{40}$$

$$\overline{\boldsymbol{A}}=\begin{bmatrix}\boldsymbol{A}_{11} & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & \boldsymbol{A}_{cc}\end{bmatrix},\ \Delta=\begin{bmatrix}0 & \cdots & \boldsymbol{A}_{1c}\\ \vdots & \ddots & \vdots\\ \boldsymbol{A}_{c1} & \cdots & 0\end{bmatrix} \tag{41}$$

基于对角矩阵 $\overline{\boldsymbol{A}}$,GCN的表示函数(式(42))可以近似分解到不同的聚类分块中(式(43)和(44)),其中 $\boldsymbol{A}'$ 表示 $\boldsymbol{A}$ 的归一化形式,

$$\boldsymbol{E}^{(l)}=\boldsymbol{A}'\sigma(\cdots\boldsymbol{A}'\sigma(\boldsymbol{A}'\boldsymbol{X}\boldsymbol{W}^{(0)})\boldsymbol{W}^{(1)}\cdots)\boldsymbol{W}^{(l-1)} \tag{42}$$

$$\approx\overline{\boldsymbol{A}}'\sigma(\cdots\overline{\boldsymbol{A}}'\sigma(\overline{\boldsymbol{A}}'\boldsymbol{X}\boldsymbol{W}^{(0)})\boldsymbol{W}^{(1)}\cdots)\boldsymbol{W}^{(l-1)} \tag{43}$$

$$=\begin{bmatrix}\overline{\boldsymbol{A}}'_{11}\sigma(\cdots\overline{\boldsymbol{A}}'_{11}\sigma(\overline{\boldsymbol{A}}'_{11}\boldsymbol{X}_1\boldsymbol{W}^{(0)})\boldsymbol{W}^{(1)}\cdots)\boldsymbol{W}^{(l-1)}\\ \vdots\\ \overline{\boldsymbol{A}}'_{cc}\sigma(\cdots\overline{\boldsymbol{A}}'_{cc}\sigma(\overline{\boldsymbol{A}}'_{cc}\boldsymbol{X}_c\boldsymbol{W}^{(0)})\boldsymbol{W}^{(1)}\cdots)\boldsymbol{W}^{(l-1)}\end{bmatrix} \tag{44}$$

此外,Cluster-GCN还通过随机组合分块来缓解式(43)中存在的边遗漏(如式(40)所示,Δ 中包含的边)和估计误差的问题.在分批训练中,每一批随机选择多个聚类分块 $\{b_{i_1},b_{i_2},\cdots,b_{i_n}\}$,而不是将单个分块作为训练数据.

RWT.考虑到Cluster-GCN在大规模图应用中聚类算法产生的时间和空间开销过大,Bai等人提出了逐层游走的训练策略RWT(Ripple Walk Training)[60].RWT通过子图采样算法实现图数据分批,在每一批次中构建图神经网络模型进行训练.RWT的采样策略综合考虑随机性和图结构的连接性,首先随机采样一部分节点作为初始子图,然后采用逐层扩张的方式从当前子图的邻居节点中进行采样并更新子图,直至子图的节点个数达到阈值.并基于GCN和GAT验证了RWT的有效性.

GraphSAINT. 基于采样的图神经网络模型通常首先构建网络模型，然后进行节点采样实现分批训练. 与此不同，GraphSAINT[61] 先进行子图采样然后基于此构建网络模型. 首先基于设定的采样器估计节点和边的采样概率，基于此在每一个训练批次中进行子图采样，并根据子图构建完整的(full-batch)GCN 模型并进行训练，最终通过反向传播来更新模型参数. 此外，GraphSAINT 还通过归一化方法来消除分批训练中的偏差，并且借助随机游走策略优化采样算法以降低分批方差.

在基于子图采样的图神经网络模型优化方面，Zeng 等人提出的 GraphSAINT 通过偏差消除和误差降低在一定程度上提高了精度；在此工作中，Zeng 等人提出了一个并行训练框架来提高模型在多核平台上的可拓展性[62]. 首先，通过子图采样算法实现图数据的分批训练. 然后，从子图采样算法和图神经网络特征传播两个方面提高了程序并行性. 通过采样器之间以及采样器内部的并行化，随着处理器数量的增加理论上取得了近线性的加速比. 在图神经网络特征传播方面，通过数据划分提高了缓存利用率并减小了通信开销. 此外，Zeng 等人还提出了运行时调度器，通过重排图神经网络操作顺序和调整分批子图进一步优化并行性能.

SHADOW-GNN. SHADOW-GNN 的提出[63] 不仅是为了应对大规模数据对图神经网络模型带来的挑战，还缓解了图神经网络层数的增加造成的过度平滑的问题. 在常规的图神经网络模型中，节点的接受区域随着图神经网络层数的增加而增加，而 SHADOW-GNN 的核心想法就是将节点的接受区域和图神经网络深度之间的关联解耦，让模型具有深层网络的表达能力又能避免接受域扩张带来的过度平滑的问题. 其做法主要是基于子图采样的策略，首先通过采样形成不同的子图，然后在子图上通过任意深度的图神经网络模型来获得节点的表示.

基于子图的采样算法总结. 基于初始图(如图 5(a)所示)，ClusterGCN(如图 5(c)所示)通过节点聚类进行子图采样，以提高子图训练中的节点利用率. RWT(如图 5(b)所示)借助随机游走策略实现逐层扩张的子图采样. Zeng 等人提出的 GraphSAINT(如图 5(d)所示)侧重于通过减小估计偏差以及不同子图间的方差，来提高基于子图采样的图神经网络模型的性能. Zeng 等人还探索了此模型在多核平台上的可拓展性. SHADOW-GNN[63] 以解耦图神经网络模型的深度和节点的邻居接受域为出发点，借助子图采样的策略，在增强了模型可拓展性的同时缓解了过度平滑的问题.

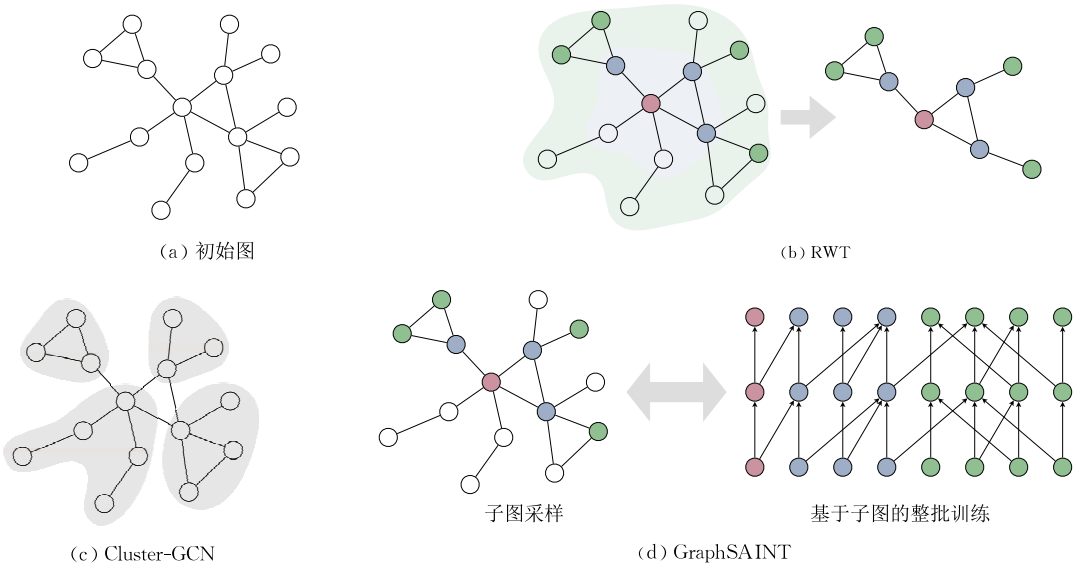

图 5　基于子图的采样算法

3.4　性能对比

Zeng 等人[61] 在 5 个公开数据集(统计信息见表 4，其中，密度表示图中边的数目与节点数目平方次之间的比值)上比较了 4 种采样算法在节点分类任务上的准确性(以 *micro_F*1 作为评价指标)，其中对比算法包括基于节点的采样算法 GraphSage[20]、基于层的采样算法 FastGCN[21] 和基于子图的采样算法 ClusterGCN[22] 和 GraphSAINT[61]，并提供了 GraphSAINT 源码以及完成预处理的数据集①. 性能对比结果如表 5 所示.

① https://github.com/GraphSAINT/GraphSAINT

表 4 数据集统计信息

数据集	节点	边	密度/%	类别	任务类型
PPI	14755	225270	0.1035	121	多标签节点分类
Flickr	89250	899756	0.0113	7	多类别节点分类
Reddit	232965	11606919	0.0214	41	多类别节点分类
Yelp	716847	6977410	0.0014	100	多标签节点分类
Amazon	1598960	132169734	0.0052	107	多标签节点分类

表 5 节点分类性能对比(评价指标为 *micro_F*1,实验数据引自文献[61])

方法	PPI	Flickr	Reddit	Yelp	Amazon
GraphSage	0.637±0.006	0.501±0.013	0.953±0.001	0.634±0.006	0.758±0.002
FastGCN	0.513±0.032	0.504±0.001	0.924±0.001	0.265±0.053	0.174±0.021
ClusterGCN	0.875±0.004	0.481±0.005	0.954±0.001	0.609±0.005	0.759±0.008
GraphSAINT	0.981±0.004	0.511±0.001	0.966±0.001	0.653±0.003	0.815±0.001

整体来说,基于子图的采样算法在不同的数据集上都有着更好的表现,*micro_F*1 指数更高且方差较小.基于层的采样算法 FastGCN 在较稀疏的大规模数据集 Yelp 和 Amazon 上分类准确性较低,其主要原因是相邻层之间的连接稀疏.GraphSage 在数据集 Flickr、Reddit、Yelp 和 Amazon 数据集上的节点分类准确性和基于子图的采样算法比较接近,但是由于其采样过程针对每个节点进行,训练过程中的节点重用率低,训练过程需要消耗更多的时间.详细的对比数据请参考文献[61].

3.5 小 结

针对图神经网络模型在大规模数据训练中存在的挑战,本节总结和分析了不同粒度的采样算法,如表 6 所示.在三类采样算法中,节点级的采样算法最先提出,其算法实现以及实现也更为直观,为大规模数据分批载入内存提供了可能,但是图神经网络中迭代更新的特征导致节点级采样算法存在邻居节点爆炸问题,在计算和内存方面都造成了一定的压力.层级的采样算法通过在每一层采样固定数目的节点,避免了邻居节点爆炸的问题,但是并行的层级采样算法存在不同层之间连接稀疏的问题;迭代式的层级采样算法能够在一定程度加强不同层之间的连接性,但运行时间较长.子图级采样算法通过生成子图并基于子图进行整批训练,在实现分批训练的同

表 6 采样算法总结

分类	算法	主要贡献	待优化方面
节点级	GraphSage[20]	·将 GNN 模型拓展到归纳式任务场景 ·图数据规则化以实现参数共享	·邻居节点爆炸问题 ·忽略节点重要性 ·估计偏差
	PinSage[54]	·基于节点重要性采样 ·聚合时进行重要性加权	·邻居节点爆炸问题 ·估计偏差
	VR-GCN[55]	·减小采样算法造成的梯度偏差 ·基于理论分析,减小了采样数量	· 邻居节点爆炸问题
	LGCL[56]	·借助特征重组将数据转化到欧氏空间 ·便于已有 CNN 算法进行优化	·节点表示过于平滑 ·邻居节点爆炸问题
层级	FastGCN[21]	·缓解邻居爆炸问题 ·不同层之间独立采样 ·基于重要性采样优化模型	·加剧图数据稀疏性 ·冗余节点计算
	AS-GCN[57]	·提高节点表示重用率 ·显式减小方差 ·借助连接跳跃捕捉二阶相似性	·采样过程并行性较低
	LADIES[58]	·基于重要性采样	·采样过程并行性较低
子图级	Cluster-GCN[22]	·提高节点表示重用率	·存在聚类算法开销 ·估计偏差
	RWT[60]	·增强随机性 ·捕捉图结构数据的连接性	·计算复杂度高
	GraphSAINT[61]	·借助归一化技术减小估计误差 ·采样过程考虑方差	·仅考虑同构图
	SHADOW-GNN[63]	·解耦深度和邻居接受域 ·缓解了过度平滑的问题	·仅考虑同构图

时最大程度上提高了节点重用率.但是采样生成的子图和原图本身分布不一致,存在采样造成的估计偏差问题.

以上基于节点、层和子图的采样算法实现了图神经网络模型的分批训练,在一定程度上缓解了大规模数据训练中存在的内存限制问题,增加了模型的可拓展性.并通过重要性采样、方差消减、随机组合等方式提高模型收敛性.但目前的采样算法主要基于静态的同构图进行优化,忽略了现实应用中图数据的异构性、动态性、幂律分布等复杂特征.

4　图神经网络框架加速

图神经网络中计算过程包含图结构数据的不规则访存(集中于聚合阶段)和复杂特征的大量计算(集中于更新阶段).然而,传统神经网络编程框架只擅长加速规则数据的计算效率,在不规则访存方面性能较差.图计算框架在图遍历等不规则访存任务上表现较好,但并不适用于节点复杂特征的访存和计算.针对以上问题,不少研究者提出了面向图神经网络模型的编程框架并进行了相关优化技术的探索,为面向大规模数据的图神经网络模型运行以及优化奠定了基础.本节将分别对编程框架和相关优化技术进行分析和总结.

4.1　图神经网络编程框架

本节将对 Deep Graph Library、PyTorch Geometric、Graph-Learn 等主流的编程框架进行总结,如表 7 所示.

表 7　图神经网络编程框架

名称	主要特征	链接
DGL[23]	• 基于 PyTorch、MXNet 和 TensorFlow • message,reduce & update functions • 支持多节点的分布式训练 • 提供大量说明文档和实现样例	https://www.dgl.ai
PyG[24]	• 基于 PyTorch • message,reduce & update functions • 支持多节点的分布式训练 • 除了图数据,支持对关系学习和 3D 数据的处理	https://pytorch-geometric.readthedocs.io
Graph-Learn[16]	• 应用于推荐系统、个性化检索等阿里巴巴商用场景 • 存储、采样、计算操作 • aggregate & combine (兼容 TensorFlow 和 Pytorch) • 支持动态图	https://github.com/alibaba/graph-learn
NeuGraph[70]	• 提出 SAGA 编程模型 • 提出图相关的优化技术 • 支持多 GPU 训练	—
FlexGraph[72]	• 提出 NAU 编程模型 • 支持层级聚合方式	—

DGL. 作为目前最受欢迎的开源图神经网络拓展库之一,Deep Graph Library(DGL)[23] 支持以深度学习库 PyTorch、MXNet 和 TensorFlow 为后端,并且提供了大量的说明文档、用户手册和实现样例以便于用户学习应用.基于消息传递机制[44,64],DGL 将图神经网络模型抽象为三个阶段,并提供对应的调用接口以便于用户实现和优化.(1)消息函数(message function)用于捕捉每一条边及其对应端点的特征,记作边信息;(2)针对每一个节点,归约函数(reduce function)对其边信息进行聚合生成中间表示,常见形式包括求和、均值、最大值、最小值、LSTM 等;(3)基于归约函数生成的中间表示,更新函数(update function)最终对节点表示进行更新.

为了提高计算效率,DGL 将消息函数和更新函数中的基本计算操作转化为稀疏矩阵运算,包括泛化的稀疏-稠密型矩阵运算(generalized Sparse-dense Matrix Multiplication,g-SpMM)和泛化的采样稠密-稠密型矩阵运算(generalized Sampled Dense-Dense Matrix Multiplication,g-SDDMM).此外,针对亿级节点或边的大规模图数据,DistDGL 提供了基于多运算节点的分布式训练模块[65].

PyG. PyTorch Geometric(PyG)[24] 是另一个常用的开源图神经网络扩展库,同样基于消息传递机制,以 PyTorch 为后端实现.在模型应用方面,用户可以通过定义消息、更新函数和选择聚合函数(求和、均值、最大值)来灵活快速地实现图神经网络算法的构建.也可以直接应用已有的图神经网络模型.在具体操作方面,PyG 提供了消息传递机制、池化、归一化以及输出函数相关操作的支持.在数据存储

和加载方面，提供了大量的常用数据集以及分批处理、节点采样、子图采样等相关的数据加载器(Data Loaders)．除了图结构数据，PyG 也支持对关系学习和 3D 数据的处理．

基于此，Pytorch-BigGraph[66] 通过图划分和支持分布式训练增大了可处理的图数据规模．Pytorch-BigGraph 支持多实体、多关系网络表示，并提供了高效的负采样算法．

Graph-Learn．Graph-Learn(原 AliGraph)[16] 是阿里巴巴构建的一个完善的开源图神经网络系统，目前已经应用于推荐系统、个性化检索等阿里巴巴的商用场景．Graph-Learn 主要包含存储、采样和计算操作三个层次的优化．在存储层次，Graph-Learn 实现了四种图划分方式以支持不同的场景需求：METIS[59]、1D 划分[67]、2D 划分[68] 和流式划分[69]，采用分布式存储，并且借助缓存来减小通信开销．在采样层次，Graph-Learn 构建了三种采样器，并提供了 Python 和 C++接口．遍历采样用于从本地子图中生成分批训练的节点和边，邻居采样用于从本地缓存中获取一跳或者多跳邻居节点，负采样一般从本地(也支持跨服务器获取)生成负样本．在计算操作层次，Graph-Learn 从图神经网络模型中抽象出两个基本操作，兼容 TensorFlow 和 Pytorch．聚合操作(aggregation)用于捕捉邻居信息以生成中间表示和组合操作(combination，类似于 DGL 中的更新操作)用于通过中间结果更新节点表示．

Graph-Learn 借助以上三个层次的优化，实现对上层算法和应用在访存和计算方面的优化，以及对大规模、异构、含属性特征和动态图数据的支持．

NeuGraph．为了支持面向图数据的大规模并行神经网络计算，微软结合图和神经网络模型提出了 NeuGraph 框架[70]．其实现基于 TensorFlow，目前尚未开源．基于分布式图计算中的 GAS(Gather-Apply-Scatter)框架[67]，NeuGraph 提出了一个以节点为中心的编程模型 SAGA-NN(Scatter-ApplyEdge-Gather-ApplyVertex with Neural Networks)．其中，ApplyEdge 和 ApplyVertex 可由用户定义，分别用于更新边和节点表示．Scatter 和 Gather 由系统自动触发，Scatter 传播节点表示到其对应的边上作为 ApplyEdge 的输入，Gather 传播边特征到目标节点并进行聚合作为 ApplyVertex 的输入．

此外，为了进一步提高图神经网络的计算效率和可拓展性，NeuGraph 在图划分、调度和程序并行方面采用了一些优化技术．NeuGraph 采用局部性感知的图划分算法，比如 Kernighan-Lin 算法[71]，将有公共节点的边尽可能划分在同一块中，以此来减小块之间的通信开销．流水线调度被用于重叠计算和传输过程，以此来减小传输延迟．为了减少 ApplyEdge 中的冗余计算，NeuGraph 将仅和节点相关的计算过程调度到前一层的 ApplyVertex 中，实现计算复杂度从 $O(|\mathcal{E}|)$ 到 $O(|\mathcal{V}|)$ 的转变，其中 $|\mathcal{E}|$ 和 $|\mathcal{V}|$ 分别表示边和节点的数量．为了增强可扩展性，NeuGraph 支持多 GPU 的并行训练．

FlexGraph．阿里巴巴和上海交通大学合作提出了分布式图神经网络训练框架 FlexGraph[72]．将现有方法中的邻居定义分为直接邻居和间接邻居，将聚合方式分为水平聚合(flat aggregation)和层级聚合(hierarchical aggregation)．并根据邻居定义和聚合方式，将现有图神经网络方法分为以下三类：考虑直接邻居的水平聚合方式[45,73-74](Direct Neighbors with Flat Aggregation，DNFA)、考虑间接邻居的水平聚合方式[54](Indirect Neighbors with Flat Aggregation，INFA)以及考虑间接邻居的层级聚合方式[75-77](Indirect Neighbors with Hierarchical Aggregation，INHA)．

FlexGraph 将图神经网络编程抽象为 NAU 三个阶段：邻居选择(neighbor selection)、聚合(aggregation)和更新(update)，并针对以上三类模型提供了对应的支持和优化．同时，指出大部分现有的图神经网络模型编程框架(包括 DGL[23]、PyG[24]、NeuGraph[70]、Euler①)仅支持属于 DNFA 分类的模型．为了进一步加速训练，FlexGraph 通过应用驱动的策略来促进负载平衡和流水线调度技术来重叠计算和通信过程．

4.2 框架相关优化技术

以上面向图神经网络的编程框架为模型实现提供了一定程度的便利，然而，在运算过程中往往存在计算效率低、内存消耗大、通信时延高、负载不均衡等问题，这些问题在面向大规模图数据训练过程中尤为显著．针对以上问题，许多研究者提出了对应的优化策略．本节将图神经网络编程框架相关的优化技术按照其优化方面分为 5 个部分，分别是数据划分、任务调度、并行执行、内存管理和其他方面，总结如表 8 所示．在每一方面相关的优化技术分析中，首先概括其优化目标及常用方法，然后逐一介绍具体的优化策略，最后进行总结．

① https://github.com/alibaba/euler

表 8　图神经网络框架相关优化技术

名称	基础框架	优化方面				
		数据划分	任务调度	并行执行	内存管理	其他
P3[78]	DGL	√	√	√	√	
G3[79]	Gunrock					√
ROC[80]	FlexFlow	√			√	
GNNAdvisor[81]	DGL,PyG,NeuGraph,Gunrock	√			√	√
FeatGraph[82]	DGL	√		√	√	
PCGCN[83]	TensorFlow	√				
Huang 等人[84]	DGL,PyG	√	√		√	√
DistDGL[65]	DGL	√				

4.2.1　数据划分

数据划分是大规模数据处理中的一项基础任务,通过将数据划分成多块,以采用分批处理的方式或者分配到分布式系统中的不同机器并行处理,来提高计算效率以及解决单个机器内存不足的问题.数据划分算法主要的优化目标是提高不同批训练数据的负载均衡和单批次数据内部的空间局部性.与典型图算法相同,针对图神经网络模型的数据划分算法需要对图结构(即节点维度和边维度)进行划分,此外还需要在特征维度进行划分.

随机划分. P^3[78] 基于图神经网络算法 GraphSage,分析了 4 种图划分策略:哈希随机划分策略[85]、均衡的最小边切割划分策略[59]、节点切割划分策略随机节点切分[86]和 GRID[67]以及最近提出的 3D 划分策略[87].发现相比于随机划分,复杂的划分策略虽然取得了一定程度的计算效率提升,但是划分过程造成了大量的计算和内存开销,并且精确划分往往只优化了一阶邻居的访存开销而无法优化高阶邻居节点.因此 P^3 采用随机哈希划分(即 1D 划分[86,88])的策略,对图结构和特征向量进行划分.

聚类感知的划分.图结构数据的稀疏性和不规律性在一定程度上降低了图卷积神经网络(GCN)在 GPU 上的计算效率,Tian 等人[83]借助实验验证了自然图存在聚类属性,并提出了 PCGCN 来优化图划分中的局部性,从而进一步提高 GCN 的运行效率.对每一个划分的子图,根据其密度选择不同的表示和计算方法.针对稀疏子图,PCGCN 采用行压缩(Compressed Sparse Row,CSR)的形式来表示子图并采用对应的稀疏矩阵乘法来进行 GCN 中的消息传播.针对密集子图,则采用密集邻接矩阵及其对应的矩阵乘法来完成运算.

节点度感知的划分.现实场景中的图数据常常存在幂律分布,比如,电商平台中大部分的交易产生于少量的热门商品,社交网络中大部分的管理和少量的热门用户有关.节点度数的幂律分布对以负载均衡为目标的图划分技术提出了更大的挑战.Huang 等人[84]通过实验分析发现现有图神经网络编程框架 DGL[23]、NeuGraph[70]等基于目标节点进行数据划分,尽管每一块数据包含的目标节点数目相对均衡,但是由于图数据的不规则性,不同块数据中包含的邻居节点数目存在较大的差异性,造成了严重的负债不均衡问题.针对此问题,Huang 等人提出通过邻居分组(neighbor grouping)策略来实现更均衡的调度.具体来说,针对目标节点,将邻居节点分组并设置组内可包含节点数目的上界,同一组内的节点被分配到一个计算单元.按此策略划分,针对度较大的邻居节点,其邻居节点被分为多组,计算任务由多个单元执行.能够改善图结构不规则性造成的负债不均衡问题,同时提高了数据的空间局部性.

结合特征维度的划分. GNNAdvisor[81]综合粗粒度邻居节点划分、细粒度的特征维度划分以及基于 warp(warp 是 GPU 上 SIMT(单指令多线程)处理器执行的基本单元,一个 warp 包含 32 个线程)的线程对齐策略,来促进负载平衡和减小访问开销.在划分过程中,FeatGraph[82]也同时考虑了图结构和节点特征,通过同时优化图结构遍历和特征向量计算来加速图神经网络运行.

动态划分.基于分布式深度学习训练框架 FlexFlow[89],Jia 等人提出了一个面向多 GPU 环境的图神经网络框架 ROC[80],主要在数据划分和内存管理两个方面进行了优化.在图数据划分方面,ROC 基于线性回归模型提出了一个动态的图划分策略,在图神经网络训练和推理阶段都可以达到负载平衡的目的.

DistDGL[65]为 DGL 提供了图神经网络模型的分布式训练支持.在数据划分方面,DistDGL 采用 METIS 算法[59]来平衡每个分块的节点数量和减小

边切割,并采用多约束机制来保证分批训练中节点以及边数量和类型的平衡性.此外,DistDGL 还支持分布式数据访存以及采样算法.

目前数据划分通常采用随机或者基于直接关联的划分策略,虽然在一定程度上实现了数据的分批处理,提升了模型效率并解决了内存限制的问题.但是图神经网络模型往往涉及图结构中的高阶关联,如何将这一需求与图数据的分布特征相结合进一步提升分批训练的效率仍然有待进一步优化.

4.2.2 任务调度

任务调度通过改变任务的执行顺序,来提高整体的运行效率.比如,在分布式系统中通过重叠计算和通信任务执行阶段来减小通信时延,通过将具有相似访存需求的任务邻近执行来提高时间局部性进而提高缓存命中率.在图神经网络中,图结构的稀疏性、不规则性以及特征的高维度都是任务调度尤其是分布式系统中的调度需要考虑的关键特征.

P^3[78]将图神经网络模型执行过程划分为顺序执行的 4 个阶段:前向传播中的模型并行和数据并行阶段、反向传播中的数据并行和模型并行阶段.其中,前向传播中,数据并行依赖于模型并行产生的结果,需要等待不同机器间的通信同步.反向传播过程也存在类似的通信延迟问题.受到 PipeDream[90]的启发,P^3对不同批次数据之间的执行过程进行调度,通过重叠计算和通信过程来减小通信延迟造成的影响.

为了提高缓存命中率,Huang 等人[84]利用局部性感知的任务调度策略,将具有相似邻居的节点相邻处理.具体来说,首先基于 Jaccard 系数计算两两节点间的相似性,然后合并高相似的节点形成聚类,最后基于聚类结果进行任务调度.除了邻居分布,节点邻居数量也存在一定的差异性,是造成负载不均衡的原因之一.Huang 等人采用邻居分组策略进行更细粒度的调度.将邻居节点均匀分成不同的组,并将每个邻居分组对应的任务分配到 warp 进行运算.

以上任务调度策略虽然在一定程度上提高了整体的运行效率,但往往针对具体的模型进行研究和测试,在提高调度策略的通用性方面有待进一步的研究.

4.2.3 并行执行

并行计算是提高系统计算效率和处理能力的常见方式,也是大规模数据应用的重要基础.根据其并行执行的层次,机器学习将并行模型分为数据并行、模型并行和混合并行三类.数据并行模型将数据划分为不同的块并分配到不同的计算单元中,每一个计算单元针对其对应的数据块执行相同的模型.而模型并行则与之相反,将模型划分为不同的部分(比如,深度学习中的不同层)并分配给计算单元,每一个计算单元对相同的数据执行其对应的模型分块.混合并行是以上二者的结合.图神经网络中迭代的执行过程以及图数据之间的依赖性使得已有的并行方式无法直接应用其中.

FeatGraph[82]分别优化聚合和更新阶段的核函数.与已有框架不同的是,可以由作者定义更新函数和并行方式.由用户定义不同级别的并行方式,节点级别、边级别和特征级别.

针对分布式系统,P^3[78]采用混合并行的方式.首先利用模型并行以避免高维特征向量造成的传输开销,每个机器在本地对节点特征进行处理,仅对特征处理之后的中间结果进行共享.然后再利用数据并行对节点表示进行传播和更新.

在并行执行方面,图数据的不规则性造成的数据依赖仍然是一个挑战,如何在确保图拓扑结构的有效特征损失可接受的前提下,尽可能减少数据依赖性,能够进一步提升并行执行的效率.

4.2.4 内存管理

一方面,图数据的大规模性、不规则性和稀疏性以及节点和边特征的复杂性都在一定程度增加了图神经网络模型的内存开销.另一方面,相比于 CPU,GPU 虽然能够为神经网络模型中的密集计算提供计算资源的支持,但往往内存更小,比如,Nvidia 1080ti 显卡的内存容量是 12 G.这二者要求图神经网络模型,尤其是基于 GPU 平台以及异构分布式系统时,采用更有效的内存管理策略来减小模型的访存开销.

数据可见范围适配策略,Huang 等人[84]通过适配不同操作间最小数据共享范围,来减小全局数据共享造成的开销.稀疏访问和冗余消除.已有方法通常是先获取邻居节点,然后进行特征转化,最后再进行聚合.特征转化需要消耗大量的计算资源,其复杂度是 $O(\mathcal{E})$.此优化策略一方面对所有节点进行特征转化,计算复杂度为 $O(N)$,另一方面通过索引稀疏访问邻居节点,最后进行聚合.这同时减小了访问和计算开销.

P^3[78]采用贪心策略缓存图结构和节点特征,充

分利用剩余内存.

在内存优化方面，GNNAdvisor[81]采用社区感知的节点重排技术来提高空间局部性，以减小访存开销. 采用基于 warp 的内存共享技术来缓解全局内存的读写开销.

ROC[80]将内存管理任务抽象为一个访存开销最小化问题，并借助迭代式的动态编程算法来确定块数据是否存入 GPU 内存. 并通过实验证明相比于 DGL，PyG 和 NeuGraph，ROC 具有更好的可拓展性.

FeatGraph[82]提出根据阈值将节点分为高度(high-degree)和低度(low-degree)节点两个部分. 考虑到 GPU 的共享内存较小，仅将高度节点载入共享内存来提高读取效率. 划分的阈值和此策略的性能密切相关，当阈值较小时，载入共享内存的节点数量越多，读取效率越高，也面临着更高的合并开销.

以上内存管理的相关工作主要从内存限制和访存效率两个方面进行了优化，不同策略在不同的应用场景中优化效果也有所差异，比如 FeatGraph 的优化策略在长尾分布的数据上优化效果更明显. 如何针对数据的具体特征，动态选择适当的优化策略以提升整体内存管理的有效性有待进一步的研究.

4.2.5　其他方面

Liu 等人提出了一个针对 GPU 环境的图神经网络框架 G^3[79]. 已有框架通常利用矩阵操作来处理图数据，造成了内存消耗和冗余计算的问题. 针对以上问题，G^3 基于 Gunrock[91]提供了一系列以图为中心的操作，在图数据管理、任务映射和负载均衡方面进行了优化. G^3 提供了 C/C++接口用于图神经网络层构建，以及图神经网络基础操作和层用于模型构建. 相比于 Pytorch 和 TensorFlow，取得了高达上百倍的加速比.

以 DGL 和 PyG 为基础框架，Huang 等人[84]借助实验分析发现了目前图神经网络框架中存在的五个主要问题：(1) 图操作中局部性较弱造成缓存命中率低，这往往和以边为中心的优化方式以及图结构本身的不规则性有关；(2) 节点度差异造成的负载不均衡问题；(3) 频繁的函数调用引发的冗余内存读取和开销问题；(4) 将神经网络操作应用到图结构存在的内存开销和冗余计算问题；(5) 特征长度变化引起的低效性，DGL 和 PyG 往往忽略特征长度将计算任务分配到固定数目的线程中. 在实际应用中，模型性能会随着图结构和特征长度发生改变，设计自动调参以达到最佳性能.

GNNAdvisor[81]是一个基于 GPU 的图神经网络加速器，能够根据输入的图神经网络模型和图数据特征在负载分配和内存管理方面进行动态优化. 此外，GNNAdvisor 构建了分析模型来确定以上优化技术中的参数.

4.3　其　他

QGTC. 量化图神经网络(Quantized Graph Neural Network，QGNN)[92-93]凭借其高鲁棒性和低计算、内存开销，受到了一定的关注. Wang 等人通过优化张量核(Tensor Core)首次提出了面向 GPU 平台的 QGNN 框架 QGTC(Quantized GNN via GPU Tensor Core). 在算法方面，QGTC 首先对数据进行了量化的低位表示(low-bit representation)，然后将图神经网络中的操作进行位分解以适应 Nvidia GPU 库提供的基于张量核的固定位运算①. 在 GPU 核函数方面，通过子图划分和分批、零元素跳过(zero-tile jumping)策略进行计算相关的优化，通过 3D 堆叠的位压缩(3D-stacked bit compression)、非零元素重用(non-zero tile reuse)策略来实现内存方面的优化. 在框架方面，QGTC 将基于位的张量表示以及运算引入到 Pytorch 框架，来实现更加方便的编程调用和增强可拓展性.

4.4　小　结

本节首先介绍了面向图神经网络的主流编程框架并对其特征进行了分析和总结，包括应用最为广泛的 DGL 和 PyG、应用于工业场景的 Graph-Learn 以及 NeuGraph 和 FlexGraph. 然后，将现有针对图神经网络框架的优化技术分为数据划分、任务调度、并行执行、内存管理以及其他方面五类，并逐类进行介绍分析. 以上面向图神经网络的编程框架及其优化技术为图神经网络模型的灵活实现和应用以及性能优化提供了支持. 并对面向大规模数据的图神经网络模型的分布式训练提供了一定程度的支持.

但是，目前面向图神经网络的编程框架及其加速技术的研究尚处在初级阶段. 编程框架应用的广泛性、对复杂图神经网络模型的支持范围以及对复杂图数据以及异构平台的支持力度有待进一步增强. 在面向大规模数据训练时，如何从编程狂减的角

① https://developer.nvidia.com/cublas

度进行优化应对图计算、神经网络模型以及数据大规模性带来的三重挑战，以提高图神经网络模型在运行效率和建模精度方面的性能，是一个亟待解决的问题.

5 总结与展望

数据大规模性是图神经网络模型在应用中面临的主要挑战之一，大量的研究工作从不同的角度进行了性能优化和加速. 针对现有综述在这一方面的空白，本文分析总结了现有图神经网络模型存在的具体挑战，并从算法模型和编程框架两个方面介绍了图神经网络在大规模数据应用中的相关进展. 下文将对面向大规模数据的图神经网络模型中存在的挑战进行总结，并总结本文在算法模型和编程框架方面的综述工作. 然后对未来相关工作进行展望.

5.1 本文总结

(1) 常见图神经网络模型及其挑战分析. 本文首先介绍了图卷积神经网络、图注意力神经网络、图循环神经网络和基于自编码器的图神经网络四种常见的图神经网络模型，对应分析了其在大规模数据应用中存在的挑战，并分类总结分析了模型层相关的挑战. 然后从算法模型和编程框架两个方面就相关研究进行了总结和分析；(2) 算法模型. 针对大规模数据在图神经网络模型训练中带来的挑战，大部分优化工作集中在采样算法方面. 根据采样粒度，本文将现有工作分为基于节点、层和子图的采样算法三类. 针对每一类采样算法，介绍相关的主要模型并进行分析. 最后进行了全面的总结和分析；(3) 编程框架. 本文首先总结了 DGL、PyG 等主流的编程框架. 然后将现有优化技术分为数据划分、任务调度、并行执行、内存管理和其他方面五类. 针对每一个类别，简要介绍了其优化目标并列举了具体的优化策略. 最后进行了全面的总结和分析.

5.2 未来工作展望

本文从模型优化和框架加速两个方面总结了面向大规模数据的图神经网络优化的相关进展，下文将从这两个方面展望未来工作，如图 6 所示.

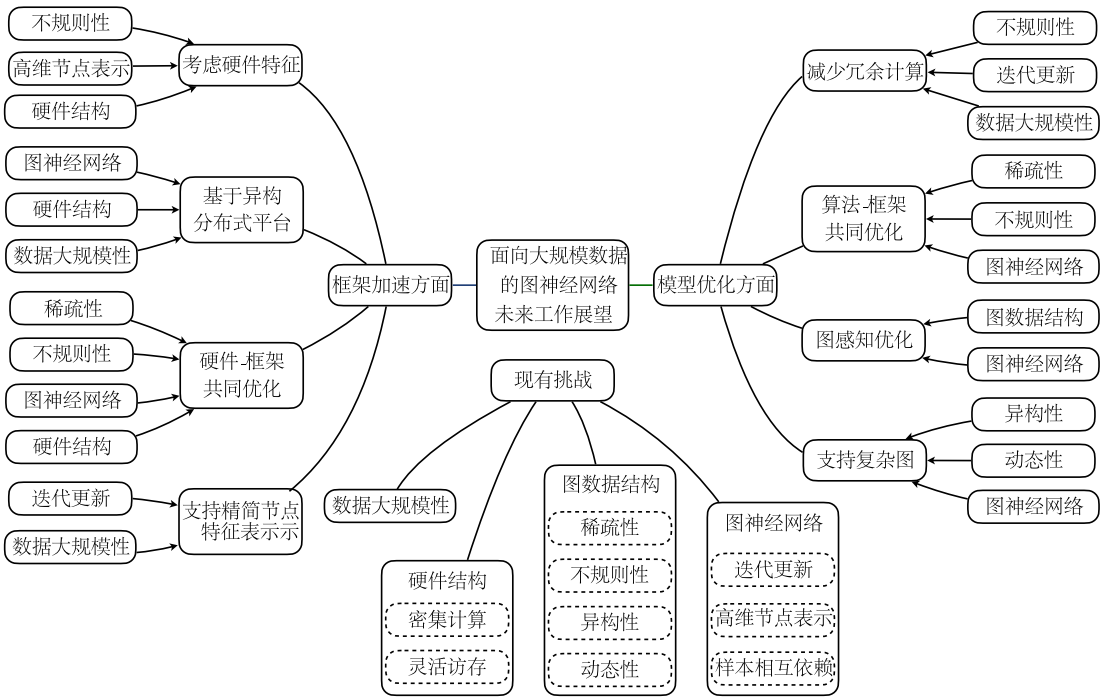

图 6 未来工作展望

模型优化方面. (1) 减少冗余计算. 图结构的不规则性以及图神经网络的迭代更新机制导致分批训练过程中存在大量的冗余计算，这一问题在大规模数据中更为显著. 如何借助理论分析和模型设计减小冗余计算，有待进一步研究；(2) 算法-框架协同优化. 目前面向图神经网络的编程框架设计处于初级阶段，其设计以通用性为首要目标，往往忽略了模型的具体特征. 如何在已有编程框架的基础上针对具体的算法特征进行个性化的优化，能够进一步提升大规模图神经网络的运行效率；(3) 图感知优化. 在

大规模图神经网络模型训练过程中，数据划分和模型训练中涉及一些超参数，根据图数据和图神经网络模型的特征自动进行参数调节，能够提高模型性能以及增强其灵活性；(4) 支持复杂图. 现有算法大多基于同构图，然而现实场景中的图数据通常包含不同类型的节点及关联[94]，关联也会动态演变[95]. 图数据的动态性、异构性等复杂特征有待在后续的研究中得到关注.

框架加速方面. (1) 考虑硬件特征的灵活调度. 针对图神经网络对访存和计算资源的具体需求，结合硬件特征，进行具体的框架优化能够达到事半功倍的效果；(2) 基于分布式异构平台的优化. 分布式平台为高效的大数据分析提供了丰富的硬件资源. 然而，现有工作大多基于单个机器，没有充分考虑大规模图神经网络在分布式异构平台上的优化；(3) 框架-硬件协同优化. 基于硬件结构的个性化优化能够从本质上满足图神经网络对不规则访存和密集计算的需求，特定的框架优化能够便于图神经网络模型的实现和应用以更有效地发挥硬件结构的功能. 二者协同优化能够达到相互促进的效果；(4) 精简节点特征及表示. 更少量的特征和更低维的表示能够显著减小计算和内存开销，并且图神经网络模型的迭代更新机制能够在一定程度上缓解精简表示造成的精度损失. 在大规模数据计算中如何平衡二者值得一定的关注.

参　考　文　献

[1] Zeng Y, Zhou X, Rao J, et al. Accurately clustering single-cell RNA-seq data by capturing structural relations between cells through graph convolutional network//Proceedings of the IEEE International Conference on Bioinformatics and Biomedicine. Seoul, Korea, 2020: 519-522

[2] Dai X, Xu F, Wang S, et al. PIKE-R2P: Protein-protein interaction network-based knowledge embedding with graph neural network for single-cell RNA to protein prediction. BMC Bioinformatics, 2021, 22(S6): 1-16

[3] Jiang H, Li L, Wang Z, et al. Graph neural network based interference estimation for device-to-device wireless communications//Proceedings of the International Joint Conference on Neural Networks. Shenzhen, China, 2021: 1-7

[4] Chen C, Li K, Teo S G, et al. Citywide traffic flow prediction based on multiple gated spatio-temporal convolutional neural networks. ACM Transactions on Knowledge Discovery from Data, 2020, 14(4): 42:1-42:23

[5] Zehmakan A N. Majority opinion diffusion in social networks: An adversarial approach//Proceedings of the AAAI Conference on Artificial Intelligence. 2021: 5611-5619

[6] Zhang Wei, Li Yang, Zhang Ji, et al. A user trajectory identification model with fusion of spatio-temporal behavior and social relation. Chinese Journal of Computers, 2021, 44(11): 2173-2188(in Chinese)
(张伟，李扬，张吉等. 融合时空行为与社交关系的用户轨迹识别模型. 计算机学报，2021，44(11)：2173-2188)

[7] Chen J, Li K, Bilal K, et al. A bi-layered parallel training architecture for large-scale convolutional neural networks. IEEE Transactions on Parallel and Distributed Systems, 2019, 30(5): 965-976

[8] Pu B, Li K, Li S, et al. Automatic fetal ultrasound standard plane recognition based on deep learning and IIoT. IEEE Transactions on Industrial Informatics, 2021, 17(11): 7771-7780

[9] Wu Yue, Wang Ying, Wang Xin, et al. Motif-based hypergraph convolution network for semi-supervised node classification on heterogeneous graph. Chinese Journal of Computers, 2021, 44(11): 2248-2260(in Chinese)
(吴越，王英，王鑫等. 基于超图卷积的异质网络半监督节点分类. 计算机学报，2021，44(11)：2248-2260)

[10] Gori M, Monfardini G, Scarselli F. A new model for learning in graph domains//Proceedings of the IEEE International Joint Conference on Neural Networks. Montreal, Canada, 2005, 2: 729-734

[11] Scarselli F, Gori M, Tsoi A C, et al. The graph neural network model. IEEE Transactions on Neural Networks, 2009, 20(1): 61-80

[12] Wu W, Li B, Luo C, et al. Hashing-accelerated graph neural networks for link prediction//Proceedings of the Web Conference. Ljubljana, Slovenia, 2021: 2910-2920

[13] He X, Deng K, Wang X, et al. LightGCN: Simplifying and powering graph convolution network for recommendation//Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval. Xi'an, China, 2020: 639-648

[14] Dong H, Chen J, Feng F, et al. On the equivalence of decoupled graph convolution network and label propagation//Proceedings of the Web Conference 2021. Ljubljana, Slovenia, 2021: 3651-3662

[15] Chen C, Li K, Teo S G, et al. Gated residual recurrent graph neural networks for traffic prediction//Proceedings of the AAAI Conference on Artificial Intelligence. Honolulu, USA, 2019: 485-492

[16] Zhu R, Zhao K, Yang H, et al. AliGraph: A comprehensive graph neural network platform. Proceedings of the VLDB Endowment, 2019, 12(12): 2094-2105

[17] Md V, Misra S, Ma G, et al. DistGNN: Scalable distributed training for large-scale graph neural networks//Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. St. Louis, USA, 2021: 76:1-76:14

[18] Thorpe J, Qiao Y, Eyolfson J, et al. Dorylus: Affordable, scalable, and accurate GNN training with distributed CPU servers and serverless threads//Proceedings of the USENIX Symposium on Operating Systems Design and Implementation. 2021: 495-514

[19] Bahri M, Bahl G, Zafeiriou S. Binary graph neural networks //Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2021: 9492-9501

[20] Hamilton W L, Ying Z, Leskovec J. Inductive representation learning on large graphs//Proceedings of the Conference on Neural Information Processing Systems. Long Beach, USA, 2017: 1024-1034

[21] Chen J, Ma T, Xiao C. FastGCN: Fast learning with graph convolutional networks via importance sampling//Proceedings of the International Conference on Learning Representations. Vancouver, Canada, 2018

[22] Chiang W L, Liu X, Si S, et al. Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks//Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Anchorage, USA, 2019: 257-266

[23] Wang M, Yu L, Zheng D, et al. Deep graph library: Towards efficient and scalable deep learning on graphs. arXiv preprint arXiv:1909.01315, 2019

[24] Fey M, Lenssen J E. Fast graph representation learning with PyTorch geometric. arXiv preprint arXiv:1903.02428, 2019

[25] Huang G, Dai G, Wang Y, et al. GE-SpMM: General-purpose sparse matrix-matrix multiplication on GPUs for graph neural networks//Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. Georgia, USA, 2020, 72: 1-12

[26] Zeng H, Prasanna V K. GraphACT: Accelerating GCN training on CPU-FPGA heterogeneous platforms//Proceedings of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. Seaside, USA, 2020: 255-265

[27] Chen C, Li K, Zou X, et al. DyGNN: Algorithm and architecture support of dynamic pruning for graph neural networks //Proceedings of the ACM/IEEE Design Automation Conference. San Francisco, USA, 2021: 1201-1206

[28] Yan M, Deng L, Hu X, et al. HyGCN: A GCN accelerator with hybrid architecture//Proceedings of the IEEE International Symposium on High Performance Computer Architecture. San Diego, USA, 2020: 15-29

[29] Wu Z, Pan S, Chen F, et al. A comprehensive survey on graph neural networks. IEEE Transactions on Neural Networks and Learning Systems, 2021, 32(1): 4-24

[30] Zhou J, Cui G, Hu S, et al. Graph neural networks: A review of methods and applications. AI Open, 2020, 1: 57-81

[31] Ma Shuai, Liu Jian-Wei, Zuo Xin. Survey on graph neural network. Journal of Computer Research and Development, 2022, 59(1): 47-80(in Chinese)
(马帅, 刘建伟, 左信. 图神经网络综述. 计算机研究与发展, 2022, 59(1): 47-80)

[32] Xu Bing-Bing, Cen Ke-Ting, Huang Jun-Jie, et al. A survey on graph convolutional neural network. Chinese Journal of Computers, 2020, 43(5): 755-780(in Chinese)
(徐冰冰, 岑科廷, 黄俊杰等. 图卷积神经网络综述. 计算机学报, 2020, 43(5): 755-780)

[33] Abadal S, Jain A, Guirado R, et al. Computing graph neural networks: A survey from algorithms to accelerators. ACM Computing Surveys, 2022, 54(9): 191:1-191:38

[34] Zhao Gang, Wang Qian-Ge, Yao Feng, et al. Survey on large-scale graph neural network systems. Journal of Software, 2022, 33(1): 150-170(in Chinese)
(赵港, 王千阁, 姚烽等. 大规模图神经网络系统综述. 软件学报, 2022, 33(1): 150-170)

[35] Li Han, Yan Ming-Yu, Lü Zheng-Yang, et al. Survey on graph neural network acceleration architectures. Journal of Computer Research and Development, 2021, 58(6): 1204-1229(in Chinese)
(李涵, 严明玉, 吕征阳等. 图神经网络加速结构综述. 计算机研究与发展, 2021, 58(6): 1204-1229)

[36] Lin H, Yan M, Ye X, et al. A comprehensive survey on distributed training of graph neural networks. arXiv preprint arXiv:2211.05368, 2022

[37] Shao Y, Li H, Gu X, et al. Distributed graph neural network training: A survey. arXiv preprint arXiv:2211.00216, 2022

[38] Besta M, Hoefler T. Parallel and distributed graph neural networks: An in-depth concurrency analysis. arXiv preprint arXiv:2205.09702, 2022

[39] Chen J, Li K, Li K, et al. Dynamic planning of bicycle stations in dockless public bicycle-sharing system using gated graph neural network. ACM Transactions on Intelligent Systems and Technology, 2021, 12(2): 25:1-25:22

[40] Chen C, Li K, Wei W, et al. Hierarchical graph neural networks for few-shot learning. IEEE Transactions on Circuits and Systems for Video Technology, 2022, 32(1): 240-252

[41] Hamilton W L. Graph representation learning. Synthesis Lectures on Artificial Intelligence and Machine Learning, 2020, 14(3): 1-159

[42] LeCun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 1998, 86(11): 2278-2324

[43] Hochreiter S, Schmidhuber J. Long short-term memory. Neural Computation, 1997, 9(8): 1735-1780

[44] Gilmer J, Schoenholz S S, Riley P F, et al. Neural message passing for quantum chemistry//Proceedings of the International Conference on Machine Learning. Sydney, Australia, 2017: 1263-1272

[45] Kipf T N, Welling M. Semi-supervised classification with graph convolutional networks//Proceedings of the International Conference on Learning Representations. Toulon, France, 2017

[46] Ma Teng-Fei. Graph Neural Network: Foundation and Frontier. Beijing: Publishing House of Electronics Industry, 2021(in Chinese)
(马腾飞. 图神经网络：基础与前沿. 北京：电子工业出版社, 2021)

[47] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need//Proceedings of the Conference on Neural Information Processing Systems. Long Beach, USA, 2017: 5998-6008

[48] Velickovic P, Cucurull G, Casanova A, et al. Graph attention networks//Proceedings of the International Conference on Learning Representations. Vancouver, Canada, 2018

[49] Cho K, van Merrienboer B, Gülçehre C, et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation//Proceedings of the Conference on Empirical Methods in Natural Language Processing. Doha, Qatar, 2014: 1724-1734

[50] Li Y, Tarlow D, Brockschmidt M, et al. Gated graph sequence neural networks//Proceedings of the International Conference on Learning Representations. San Juan, Puerto Rico, 2016

[51] Kramer M A. Nonlinear principal component analysis using autoassociative neural networks. AIChE Journal, 1991, 37(2): 233-243

[52] Goodfellow I J, Bengio Y, Courville A C. Deep Learning. USA: MIT Press, 2016

[53] Wang D, Cui P, Zhu W. Structural deep network embedding//Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. San Francisco, USA, 2016: 1225-1234

[54] Ying R, He R, Chen K, et al. Graph convolutional neural networks for web-scale recommender systems//Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. London, UK, 2018: 974-983

[55] Chen J, Zhu J, Song L. Stochastic training of graph convolutional networks with variance reduction//Proceedings of the International Conference on Machine Learning. Stockholmsmässan, Sweden, 2018: 941-949

[56] Gao H, Wang Z, Ji S. Large-scale learnable graph convolutional networks//Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining. London, UK, 2018: 1416-1424

[57] Huang W B, Zhang T, Rong Y, et al. Adaptive sampling towards fast graph representation learning//Proceedings of the Conference on Neural Information Processing Systems. Montréal, Canada, 2018: 4563-4572

[58] Zou D, Hu Z, Wang Y, et al. Layer-dependent importance sampling for training deep and large graph convolutional networks//Proceedings of the Conference on Neural Information Processing Systems. Vancouver, Canada, 2019: 11247-11256

[59] Karypis G, Kumar V. METIS: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices. Technical Report, Department of Computer Science and Engineering, University of Minnesota, Minneapolis, USA, 97-061, 1997

[60] Bai J, Ren Y, Zhang J. Ripple walk training: A subgraph-based training framework for large and deep graph neural network//Proceedings of the International Joint Conference on Neural Networks. Shenzhen, China, 2021: 1-8

[61] Zeng H, Zhou H, Srivastava A, et al. GraphSAINT: Graph sampling based inductive learning method//Proceedings of the International Conference on Learning Representations. Addis Ababa, Ethiopia, 2020

[62] Zeng H, Zhou H, Srivastava A, et al. Accurate, efficient and scalable training of graph neural networks. Journal of Parallel and Distributed Computing, 2021, 147: 166-183

[63] Zeng H, Zhang M, Xia Y, et al. Decoupling the depth and scope of graph neural networks//Proceedings of the Conference on Neural Information Processing Systems. 2021: 19665-19679

[64] Battaglia P W, Hamrick J B, Bapst V, et al. Relational inductive biases, deep learning, and graph networks. arXiv preprint arXiv:1806.01261, 2018

[65] Zheng D, Ma C, Wang M, et al. DistDGL: Distributed graph neural network training for billion-scale graphs//Proceedings of the 10th IEEE/ACM Workshop on Irregular Applications: Architectures and Algorithms. Atlanta, USA, 2020: 36-44

[66] Lerer A, Wu L, Shen J, et al. PyTorch-BigGraph: A large scale graph embedding system//Proceedings of the Machine Learning and Systems. Stanford, USA, 2019

[67] Gonzalez J E, Low Y, Gu H, et al. Powergraph: Distributed graph-parallel computation on natural graphs//Proceedings of the USENIX Symposium on Operating Systems Design and Implementation. Hollywood, USA, 2012: 17-30

[68] Boman E G, Devine K D, Rajamanickam S. Scalable matrix computations on large scale-free graphs using 2D graph partitioning//Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. Denver, USA, 2013: 50:1-50:12

[69] Stanton I, Kliot G. Streaming graph partitioning for large distributed graphs//Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Beijing, China, 2012: 1222-1230

[70] Ma L, Yang Z, Miao Y, et al. NeuGraph: Parallel deep neural network computation on large graphs//Proceedings of the USENIX Annual Technical Conference. Renton, USA, 2019: 443-458

[71] Kernighan B W, Lin S. An efficient heuristic procedure for partitioning graphs. The Bell System Technical Journal, 1970, 49(2): 291-307

[72] Wang L, Yin Q, Tian C, et al. FlexGraph: A flexible and efficient distributed framework for GNN training//Proceedings of the European Conference on Computer Systems. 2021: 67-82

[73] Xu K, Hu W, Leskovec J, et al. How powerful are graph neural networks?//Proceedings of the International Conference on Learning Representations. New Orleans, USA, 2019

[74] Marcheggiani D, Titov I. Encoding sentences with graph convolutional networks for semantic role labeling//Proceedings of the Conference on Empirical Methods in Natural Language Processing. Copenhagen, Denmark, 2017: 1506-1515

[75] You J, Ying R, Leskovec J. Position-aware graph neural networks//Proceedings of the International Conference on Machine Learning. Long Beach, USA, 2019: 7134-7143

[76] Fu X, Zhang J, Meng Z, et al. MAGNN: Metapath aggregated graph neural network for heterogeneous graph embedding//Proceedings of the Web Conference 2020. Taipei, China, 2020: 2331-2341

[77] Xu K, Li C, Tian Y, et al. Representation learning on graphs with jumping knowledge networks//International Conference on Machine Learning. Stockholmsmässan, Sweden, 2018: 5449-5458

[78] Gandhi S, Iyer A P. P^3: Distributed deep graph learning at scale//Proceedings of the USENIX Symposium on Operating Systems Design and Implementation. 2021: 551-568

[79] Liu H, Lu S, Chen X, et al. G^3: When graph neural networks meet parallel graph processing systems on GPUs. Proceedings of the VLDB Endowment, 2020, 13(12): 2813-2816

[80] Jia Z, Lin S, Gao M, et al. Improving the accuracy, scalability, and performance of graph neural networks with ROC//Proceedings of the Machine Learning and Systems 2020. Austin, USA, 2020

[81] Wang Y, Feng B, Li G, et al. GNNAdvisor: An adaptive and efficient runtime system for GNN acceleration on GPUs //Proceedings of the USENIX Symposium on Operating Systems Design and Implementation. 2021: 515-531

[82] Hu Y, Ye Z, Wang M, et al. FeatGraph: A flexible and efficient backend for graph neural network systems//Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. Georgia, USA, 2020, 71: 1-13

[83] Tian C, Ma L, Yang Z, et al. PCGCN: Partition-centric processing for accelerating graph convolutional network//Proceedings of the IEEE International Parallel and Distributed Processing Symposium. New Orleans, USA, 2020: 936-945

[84] Huang K, Zhai J, Zheng Z, et al. Understanding and bridging the gaps in current GNN performance optimizations//Proceedings of the ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. 2021: 119-132

[85] Malewicz G, Austern M H, Bik A J C, et al. Pregel: A system for large-scale graph processing//Proceedings of the ACM SIGMOD International Conference on Management of Data. Indianapolis, USA, 2010: 135-146

[86] Gonzalez J E, Xin R S, Dave A, et al. Graphx: Graph processing in a distributed dataflow framework//Proceedings of the USENIX Symposium on Operating Systems Design and Implementation. Broomfield, USA, 2014: 599-613

[87] Zhang M, Wu Y, Chen K, et al. Exploring the hidden dimension in graph processing//Proceedings of the USENIX Symposium on Operating Systems Design and Implementation. Savannah, USA, 2016: 285-300

[88] Zaharia M, Chowdhury M, Das T, et al. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing//Proceedings of the USENIX Symposium on Networked Systems Design and Implementation. San Jose, USA, 2012: 15-28

[89] Lu W, Yan G, Li J, et al. FlexFlow: A flexible dataflow accelerator architecture for convolutional neural networks//Proceedings of the IEEE International Symposium on High Performance Computer Architecture. Austin, USA, 2017: 553-564

[90] Narayanan D, Harlap A, Phanishayee A, et al. PipeDream: Generalized pipeline parallelism for DNN training//Proceedings of the ACM Symposium on Operating Systems Principles. Huntsville, Canada, 2019: 1-15

[91] Wang Y, Davidson A A, Pan Y, et al. Gunrock: A high-performance graph processing library on the GPU//Proceedings of the ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. Barcelona, Spain, 2016: 11:1-11:12

[92] Feng B, Wang Y, Li X, et al. SGQuant: Squeezing the last bit on graph neural networks with specialized quantization//Proceedings of the IEEE International Conference on Tools with Artificial Intelligence. Baltimore, USA, 2020: 1044-1052

[93] Tailor S A, Fernández-Marqués J, Lane N D. Degree-Quant: Quantization-aware training for graph neural networks//Proceedings of the International Conference on Learning Representations. 2021

[94] Zhao J, Wang X, Shi C, et al. Heterogeneous graph structure learning for graph neural networks//Proceedings of the AAAI Conference on Artificial Intelligence. 2021: 4697-4705

[95] Gabert K, Sancak K, Özkaya M Y, et al. EIGA: Elastic and scalable dynamic graph analysis//Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. St. Louis, USA, 2021: 50:1-50:15

XIAO Guo-Qing, Ph. D., associate professor. His research interests include high-performance computing, parallel and distributed processing, artificial intelligence and big data computing.

LI Xue-Qi, Ph. D. candidate. Her research interests include large-scale GNNs and recommender systems.

CHEN Yue-Dan, Ph. D., associate professor. Her research interests include high-performance computing, parallel and distributed processing.

TANG Zhuo, Ph. D., professor. His research interests include supercomputing and cloud computing, high-performance computing and artificial intelligence fusion computing.

JIANG Wen-Jun, Ph. D., professor. Her research interests include social network analysis, recommender systems.

LI Ken-Li, Ph. D., professor. His research interests include high-performance computing system software and applications.

Background

Graph Neural Network has been employed by large numbers of applications (e. g., recommender systems, link prediction, and traffic prediction), to generated embeddings containing both graph structure information and node features. When applying to large-scale data, it faces several challenges: irregular graph structures, complex node features, and dependent training samples. It puts pressure on computation efficiency, memory management, as well as the communication cost of distributed computing. To address these limitations, lots of researchers make optimizations in terms of application models, algorithm models, programming frameworks, and hardware design. In this paper, we research and summarize the related works of GNN basics, the algorithm optimization and framework acceleration for large-scale GNN models. First, we briefly overview the message passing mechanism, classify GNN models, and analyze their challenges in processing large-scale data. Second, we classify and analyze GNN algorithms for large-scale data, mainly including sampling methods at different granularities. Third, on the side of framework acceleration, we introduce mainstream programming frameworks for GNN models as well as classify and analyze optimization techniques. Fourth, we give the prospect of future work for large-scale GNN.

This work has been supported in part by the Key-Area R&D Program of Guangdong Province (2021B0101190004), the Programs of the National Natural Science Foundation of China (62172157, 62202149), the Programs of the Hunan Province (2023GK2002, 2021RC3062), the Programs of the Guangdong Province and Shenzhen (2023A1515012915, JCYJ20210324135409026), and the Program of Zhejiang Lab (2022RC0AB03).