https://crad.ict.ac.cn
62(1): 90−103, 2025

面向分布式图计算的图划分技术综述

尚俊霖　张振宇　屈稳稳　王晓玲
（华东师范大学计算机科学与技术学院　上海　200062）
（jlshang@stu.ecnu.edu.cn; ）

Survey of Graph Partitioning Techniques for Distributed Graph Computing

Shang Junlin, Zhang Zhenyu, Qu Wenwen, and Wang Xiaoling
（*School of Computer Science and Technology*, *East China Normal University*, *Shanghai* 200062）

Abstract　The graph data structure, which is adept at encapsulating intricate relationships among entities, has been widely used in a vast array of application scenarios. With the incessant progression of Internet applications and the concomitant surge in data scales, distributed graph computing systems have demonstrated superior performance compared with traditional single-machine systems in various aspects, including computational efficiency and resource scheduling. In recent years, the increasing demand for distributed graph computing systems designed for handling large-scale graph data has brought graph partitioning technology to the forefront of academic research. Based on a comprehensive analysis of graph partitioning techniques for distributed graph computing, we explain the technological backdrop of graph partitioning in these systems. We provide definitions for key concepts related to graph partitioning in modern distributed graph computing systems and present a classification scheme for existing computational models, offering insights into the current status of distributed graph computing paradigms. Subsequent sections delve into the complexities of different graph partitioning methodologies, conducting a thorough analysis to determine their respective strengths and weaknesses within the context of various distributed graph computing frameworks. Finally, we discuss the current challenges and future research directions of graph partitioning technology in distributed graph computing systems.

Key words　graph partitioning; graph data analysis and management; graph computing; distributed graph system; hypergraph partitioning

摘　要　图结构作为表达事物之间复杂关联的数据结构，被广泛使用在多种应用场景中. 随着互联网应用的不断发展，数据规模的不断增加，分布式的图计算系统相较于传统单机系统从运算时间、资源调度等各个方面显现出优越的性能. 近年来，基于大规模图数据的分布式图计算系统使用需求快速增加，图数据划分技术受到了学术界的广泛关注. 通过对分布式图计算系统中的图划分技术的研究，首先介绍了面向分布式图计算的图划分的技术背景，给出当前分布式图计算系统中的图划分相关概念的定义以及相关计算模型的分类体系，报告了分布式图计算模型的发展现状. 接着对不同的图划分策略中的具体技术进行介绍，通过在不同策略之间进行分析与比较，总结其在各类分布式图计算系统中的优势与不足. 最后就分布式图计算系统中图划分技术的发展现状，讨论了其当前存在的挑战与未来的研究方向.

关键词　图划分；图数据分析与管理；图计算；分布式图系统；超图划分

收稿日期：2023-10-10；**修回日期**：2024-02-02

基金项目：国家重点研发计划项目 (2021YFC3340702)；国家自然科学基金重点项目 (62136002)；国家自然科学基金项目 (61972155)
This work was supported by the National Key Research and Development Program of China (2021YFC3340702), the Key Program of the National Natural Science Foundation of China(62136002), and the National Natural Science Foundation of China (61972155).

通信作者：王晓玲（xlwang@cs.ecnu.edu.cn）

中图法分类号 TP391

DOI: 10.7544/issn1000-1239.202330790 CSTR: 32373.14.issn1000-1239.202330790

图作为一种基本的数据结构，被普遍用于表达数据之间复杂的关联关系. 与树、表等其他的数据结构相比，图结构能更好地存储和表达实体及其联系. 图分析被广泛应用于社交网络、网络分析、自然语言处理以及推荐系统等领域，具有重要的研究价值. 例如：电商支付平台通过对社交网络中的异常顶点进行分析和识别，提前进行风险预警，可以避免重大的财产损失；在芯片研发领域，芯片设计公司需要识别出不同组件之间的耦合性质进行对应的电路图设计，提高芯片的计算性能和功耗表现；生物科学家通过子图匹配、频繁子图挖掘等算法，识别出蛋白质的共性表达，进行疫苗研发.

随着信息技术的快速发展，图数据的规模也越来越大，传统的单机图系统难以有效地处理超大规模的图数据. 为了处理急剧膨胀的图数据规模，学术界和工业界设计实现了许多分布式图计算系统对大规模图数据进行处理分析，如 Pregel[1], GraphLab[2], PowerGraph[3], GraphX[4] 等. 然而，分布式图计算系统面临的首要挑战是如何高效地对图数据进行划分以实现并行计算. 因此，图划分在大规模图数据处理中起着至关重要的作用. 一个设计良好的图划分算法需要考虑子图计算任务的负载均衡及通信开销，从而最大程度地提高图计算的并行能力.

面向分布式图计算的图划分技术按照划分单元可以分为点划分（vertex partitioning）和边划分（edge partitioning）两个维度. 点划分将整个图的顶点集划分为不相交的子集，这意味着每个顶点只分配给 1 个分区. 这种图划分策略广泛应用于许多早期系统，如 Pregel[1] 和 GraphLab[2]. 然而，由于现实世界中图网络的顶点度数符合幂律分布，图网络中通常包含部分度数较大的顶点（具有百万个邻居的顶点）. 在点划分中，图计算系统需要处理每个顶点的所有邻居，这将导致计算负载严重失衡. 为了解决这个问题，一些分布式图并行计算系统，如 PowerGraph[3] 和 GraphX[4]，采用了边划分的策略对图进行划分. 与点划分不同，边划分将图的边集划分为不相交的子集，这意味着每个边只分配给某个唯一的分区. 在这种模式下，上述提到的高度数顶点的计算负载被划分到不同的计算节点中，保证了整体系统的负载均衡. 然而，由于边划分需要创建顶点的副本，并且副本之间需要进行状态信息的同步，过高的副本率会增加计算节点之间的通信成本，影响并行计算的效率.

图划分技术同样可以从划分模式进行分类，主要分为离线划分（offline graph partitioning）和在线划分（online graph partitioning）. 在许多早期的图划分研究中，处理的图数据规模通常较小，整张图可以在一定的时间代价下统一读取到内存中再进行划分，这就是传统的离线划分策略. 但随着图数据规模的不断扩大，这种传统的图划分策略对时间、内存等资源消耗急剧增长，出现了流式的在线划分策略. 在线划分策略每次仅需要处理部分数据，避免将整张图读入内存，能够显著降低划分资源的开销. 由于在线划分不依赖完整的图结构信息，图数据规模的大小不再成为划分算法的阻碍. 在线划分由于算法缺乏图的完整信息，导致了划分质量的损失. 点划分中的在线图划分是以 1 个点作为读取单位，边划分中的在线划分则是以 1 条边作为基本单位，二者的具体操作过程如图 1 所示.

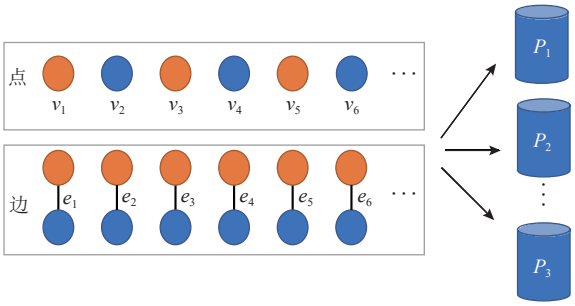

Fig. 1 Two types of graph partitioning: vertex partitioning and edge partitioning

图 1 图划分的 2 类：点划分、边划分

本文从当前流行的分布式图计算模型开始，详细调研了当前图划分技术的发展与现状. 与已有的综述类工作不同，本文不仅对经典的图划分算法如 FM 和 METIS 进行了介绍，而且针对近十年最新的图划分工作进行了详尽的调研. 不仅如此，本文同样对当前超图划分问题进行总结，并对机器学习模型在图划分问题上的应用进行讨论. 本文希望通过对图划分方法的介绍，发现现有工作的不足并对未来工作进行启发.

1 分布式图计算模型

随着计算机软硬件技术的发展，分布式图计算

系统为大规模图数据分析提供了新的技术支撑. 系统将图划分为互不相交的子集(分区), 通过并行计算进而提高图计算任务的运行效率. 对于给定图$G(V,E)$, 其中V表示图G中顶点的集合, E代表图G的边集. 图划分是将图G划分为k个子集. 用P表示划分得到的分区集合, 每一个分区用P_i表示, 其中$P_i \in P$且$\{\cup P_i = G, 0 < i < k\}$.

图划分的质量严重影响着分布式图计算系统的并行效率, 负载均衡和通信开销是衡量图划分质量最重要的2个因素. 负载均衡指的是划分得到的分区内所包含的顶点或边的数量应当基本一致, 避免计算开销产生严重倾斜. 降低通信开销则意味着需要最小化跨越分区的顶点或边的数量, 尽可能减少跨机器的网络开销. 当前图划分算法体系如图2所示.

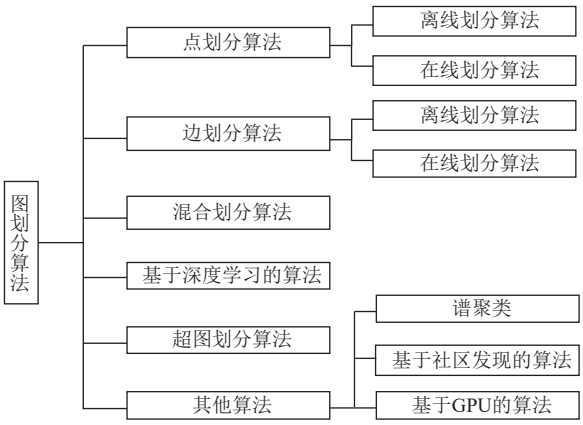

Fig. 2 Classification system of graph partitioning algorithms

图2 图划分算法分类体系

在各个系统中, 图数据的组织方式通常包括邻接列表(adjacency list)和压缩稀疏矩阵(compressed sparse row, CSR)两种方式. 邻接列表是图的一种常用表示方式, 对于每一个顶点, 都有一个与之关联的列表, 这个列表中包含了与该点相邻的所有顶点, 其主要特点是能够很方便地检查2个顶点之间是否存在关联的边, 且能够方便地获取与给定顶点相邻的所有顶点. CSR是一种压缩的稀疏矩阵格式, 它主要由3个数组构成: 非零元素数组、列索引数组以及行指针数组, 其主要特点是能够很好地节省存储空间并且提高矩阵向量运算的性能. 除此之外, SNAP同样是一种图划分算法中常用的图数据格式, 其主要特点是文件以行为单位, 每一行存放1条边的信息. 这种数据格式的特点是能够很好地节省图数据存放所需要的空间, 并且能够以边的形式动态更新图的信息.

分布式图计算系统对图进行划分时, 主要采用2种不同的划分策略: 点划分策略和边划分策略, 其各自的划分模式如图3所示. 本节将从不同计算模型采用的划分策略出发, 分别介绍点划分策略和边划分策略以及它们之间的关系.

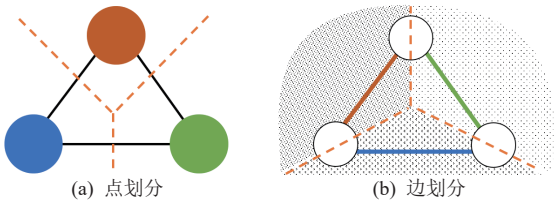

Fig. 3 Pattern diagram of vertex partitioning and edge partitioning

图3 点划分和边划分模式图

1.1 以顶点为粒度的计算模型

点划分策略是图划分问题中经典的策略之一, 以顶点为中心的图计算系统同样是研究最为广泛的系统之一. 具体而言, 给定图$G=(V,E)$, 点划分策略将图中的顶点集合V划分到不同的分区$P=(V(P),E(P))$, 同时每个分区中的顶点集$V(P)$互斥. 在这种分区方式中, 每一条跨越分区的边被称为切边(cut). 对于分布式图计算系统而言, 跨越分区的边会产生通信需求. 因此, 点划分的主要目标是在保证负载均衡的前提下最小化跨越分区的边数.

具体而言, 切边[5]的定义如下:

$$e_{\text{cut}} = (u,v), u \in P_i \wedge v \in P_j, i \neq j,$$

其中u,v表示图中的顶点. 在此基础之上, 可以建模点划分中主要的优化目标为:

$$\min|\{e_{\text{cut}}\}|, \max_{P_i \in P}|V(P_i)| < (1+\gamma)\frac{|V(P_i)|}{|P|},$$

其中γ表示平衡系数[5].

很多早期的分布式图计算系统如Pregel[1], GraphLab[2]等系统都采用了点划分策略和以顶点为粒度的计算模型. 以Pregel为例, 其通过迭代的方式处理计算任务, 每一次迭代都对应着BSP (bulk synchronous parallel)模型中的一个环节. Pregel提供了用户自定义函数*compute*(*msgs*), 允许用户定义图中顶点需要执行的运算逻辑. 其中*msgs*代表上次迭代中发送给顶点的消息集合. Pregel通过*compute*(·)函数更新顶点的状态、发送消息给其他顶点、投票终止程序等步骤实现以顶点为粒度的图计算任务.

后续的一些工作对Pregel系统进行了进一步的优化及实现. 例如Giraph[6]采用了多线程架构下的超步拆分技术, 实现对较大的消息进行拆分, 是目前使用最多的Pregel类系统; GraphLab[2]对边界顶点建立副本并采用异步计算模型, 从而消除了网络开销;

Giraph++[7]和 Blogel[8]将相邻的顶点划分到同一分区，降低了跨机器的通信开销；Mizan[9]使用了动态的顶点迁移策略以提高负载均衡；GraphD[10]将邻接表和消息存储在磁盘中，提高了系统所能处理的图数据的规模.

Pregel 类系统的关键特性在于采用消息传递模型进行迭代计算. 以常见图计算算法 PageRank 为例. 设 $pr(v)$为顶点 v的得分，初始化时，每个顶点会根据全局顶点总数初始化自身得分，即

$$pr(v) = 1/|V|.$$

在迭代中，每个顶点会将自身得分通过广播的方式发送给周围邻居，其邻居顶点会将收到的所有信息汇总，更新自身得分，并再次广播. 更新得分时，顶点会根据用户所设置的阈值，对收到的分数按照比例加权. 当所有顶点的 PageRank 值都不再发生变化，每个顶点就会投票终止计算，并结束迭代计算.

从以上过程中能够看出，以顶点为粒度的计算模型的计算开销与顶点收发的消息量显著相关. 为了保证这类计算模型的负载均衡，点划分的关键优化目标在于如何划分输入图中的高度数顶点，消除木桶效应以提高并行执行效率. 同时，由于以顶点为粒度的计算模型产生了跨机器的通信，通信瓶颈很容易制约系统的整体性能. 因此，如何降低图划分的割边数以及减少跨机器的通信开销是点划分的另一个优化目标.

1.2 以边为粒度的计算模型

与点划分策略不同，边划分策略主要对图中的边进行分区操作. 对于给定的图 $G(V,E)$，边划分策略将图中的边集合 E划分到不同的分区，并且每个分区中的边集合 $E(P)$互斥. 由于分布式图计算系统的负载主要与图中的边相关，因此相比于点划分，边划分策略可以更加均匀地将计算负载下放到不同的分区中，从而实现更好的负载均衡. 边划分策略中允许顶点跨越多个分区，从而产生多个顶点副本，因此其优化目标与点划分策略不同. 在每个顶点可能产生多个副本的情况下，每次顶点状态的改变都需要同步到其他的副本中，其存储和通信开销与每个顶点的副本个数呈正相关. 因此，边划分策略主要目标是在保证分区划分均衡的前提下，尽可能较少每个顶点的副本个数(replication factor , RF)，记作 R_{RF}，从而进一步降低图计算的存储和通信开销.

具体而言，副本个数 R_{RF} 的定义为：

$$R_{\mathrm{RF}} = \frac{1}{|V|}\sum_{P_i \in P}|V(P_i)|.$$

基于此，边划分的优化目标定义为：

$$\min \frac{1}{|V|}\sum_{P_i \in P}|V(P_i)|,$$

$$\max_{P_i \in P}|V(P_i)| < (1+\gamma)\frac{|V(P_i)|}{|P|}.$$

边划分策略最早是由 PowerGraph[3]提出，很快便被其他系统如 GraphX[4]采用. PowerGraph 提出现实世界中图的顶点度数基本符合幂律分布，因此通常包含少数度数非常大的顶点. 与之相反，图中大多数顶点仅具有几个邻居. 微博社交网络图是典型的幂律图，著名博主通常拥有上万的粉丝群体，而大部分用户的粉丝数不会超过 20 人. 在以顶点为粒度的计算模型如 Pregel 和 GraphLab 中，高度数顶点由于具有更多邻居，往往需要更多的通信时间，这导致了不同顶点存在计算负载和通信负载的巨大差异，点划分会导致工作负载分布不平衡. 而在边划分中，高度数顶点的边被分配到多个计算节点中. 因此，以边作为划分和计算的基本单位可以很好地解决负载不平衡问题，从而提高分布式图计算系统的并行效率. 在边划分中，顶点的边被划分到不同的计算节点上并存储. 集群中存在顶点的多个副本，这些副本的其中一个被指定为主(master)顶点，其他顶点被称为镜像(mirror)顶点.

基于边划分策略，PowerGraph[3]提出了聚合-应用-扩散(gather-apply-scatter, GAS)模型，该模型将计算逻辑与每个顶点的邻边相关联. 用户需要定义模型中聚合、应用以及扩散 3 个操作的具体动作，从而完成信息的传递与更新. 聚合(gather)函数的主要目的是收集信息，例如在分布式图计算中，顶点 v会从其邻居顶点收集信息. 由于每个顶点可能涉及多个副本顶点，因此还需要将所有镜像顶点的信息在主顶点上进行累加，收集到图中完整的信息. 应用(apply)函数会根据聚合函数得到的信息，对顶点自身的状态进行更新，这步操作通常是一种本地计算形式. 扩散(scatter)函数是聚合函数的逆操作. 在这个操作中，顶点会向周围邻居进行广播，更新邻居顶点的状态，用于下一轮的聚合操作.

以 PageRank 为例，GAS 在每个函数中的行为如下. 聚合函数会返回顶点 v的邻居的状态，例如邻居 u的状态：$pr(u)/d_{\mathrm{out}}(v)$. 在收集到所有信息后，顶点 v对所有状态分进行累加，通过应用函数计算当前 v的状态得分. 如果当前顶点状态更新不变或小于用户所设置的更新差值，则结束迭代，否则将会调用扩散函数更新全局状态，并进行下一轮迭代.

在GAS模型中, 当顶点v执行计算时, 由于存在顶点的镜像, 边界顶点可以直接从镜像顶点中获取邻居的值. 此时通信开销主要在于镜像顶点和主顶点之间的信息同步, 这也是边划分能够通过最小化顶点的副本数以减少通信开销的主要原因. 在负载平衡方面, 边划分需要平衡位于不同计算节点中的边的数量, 不同于点划分中对顶点数量的平衡.

GraphChi[11]是PowerGraph对应的单机版本, 它保留了GAS编程模型, 但是省去了对大内存空间机器集群的需求. 然而, 受限于单机系统的I/O带宽, 在处理大图时GraphChi比分布式系统通常要慢1个数量级. 在GraphChi的基础上, 李金吉等人[12]提出了一种流式处理的异步计算系统, 结合线程优化、优先级调度等技术提高了图计算的效率.

类似GraphChi, X-Stream[13]也是采用GAS编程模型的单机系统, 主要将顶点和顶点状态常驻在内存中, 将边保留在磁盘中, 通过数据交换在一定程度上解决了带宽限制. 当内存不足以保存所有顶点时, X-Stream将顶点划分为p个不相交的分段, 使分段p_i中的所有顶点都能放入内存, 并构成一个顶点分区. X-Stream对完全无序的边列表使用流式传输, 避免对边的随机访问.

VENUS[14]是一种分布式图计算系统, 其主要采用面向值域的编程模型. 该模型与GAS模型不同, 允许顶点v访问邻居u的状态D_u. 例如, 在PageRank算法中, 当顶点v累加每个邻居u的贡献值$pr(u)/d_{\text{out}}(u)$以计算D_v时, VENUS允许直接访问顶点u的值, 从而可以节省由写边所带来的磁盘I/O开销.

除VENUS外, Chaos[15]是一种基于X-Stream分布式图处理系统, 主要目的是解决X-Stream单机系统在处理大规模数据时面临次级存储带宽上限的问题. Photon[16]是一种按属性存放数据的分布式系统, 在保证通信开销的同时提升本地节点的计算性能. PDSim[17]是一个更具扩展性的分布式系统, 通过对分布式数据结构的重新设计实现了更细粒度的并行. cuGraph[18]是一个同时兼容点、边2种计算粒度的分布式图计算系统, 能够支持任务在计算节点的GPU中计算, 实现了分布式下对GPU硬件资源的使用.

1.3 小 结

计算模型决定了并行图计算的效率. 在并行图计算模型研究中, 研究者从MapReduce和Spark等通用的分布式计算平台出发, 设计了多种和图计算相关的特定图计算平台, 如Pregel[1], GraphLab[2]等. 然而, 这些并行图计算平台通常是针对某些特定算法设计的, 例如以点为计算中心的图计算平台可以很好地处理简单的PageRank等问题, 而GAS等以边为中心的图计算平台则可以用来解决幂律图划分的负载平衡问题. 这些不同的图计算平台很好地丰富了图计算任务的生态. 本文所介绍的划分算法按照技术发展脉络进行阐述, 如图4所示.

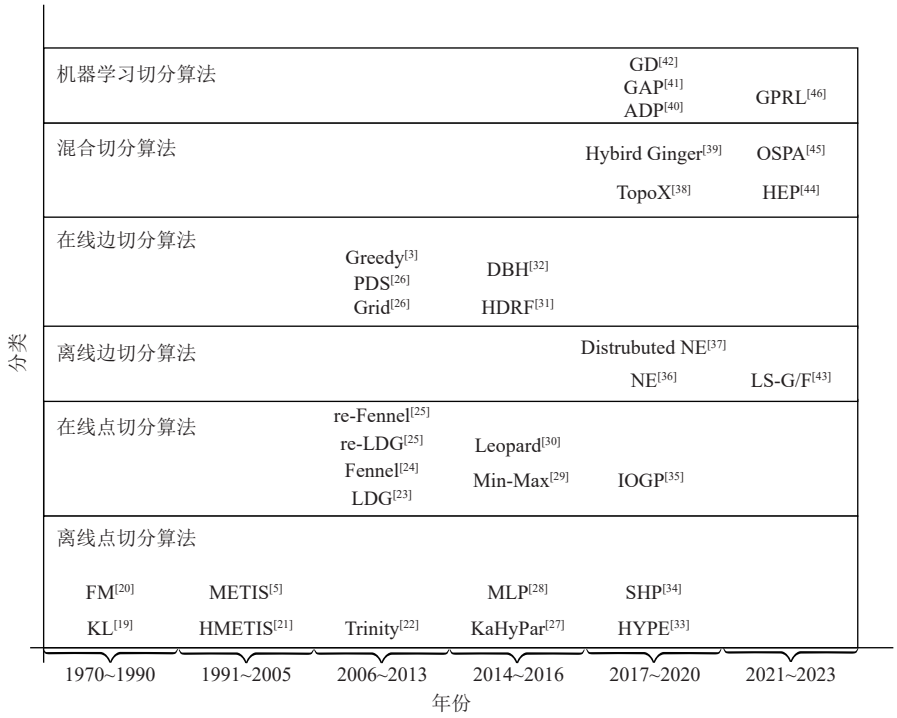

Fig. 4 Evolutionary timeline diagram of graph partitioning techniques

图4 图划分技术时间演化图

2 点划分策略

点划分策略将图划分成多个子图，并且每个子图包含互不相交的点集. 这类工作中较为经典的算法包括 KL[19], FM[20], METIS[5] 等离线划分策略，其主要特点策略会将要处理的图数据统一加载至内存中进行划分. 随着图规模的增加，传统的离线划分方法需要较高的内存占用和较长的划分时间，因此更多研究开始关注以在线流处理的方式执行图划分. 在线划分方法无需在内存中存储完整的图，而是依次读取图中的一个顶点或一条边进行划分处理. 总体而言，由于缺乏全局信息，在线划分方法的质量往往不如离线划分方法，但是划分耗时更短、扩展性更好，因此更受分布式图计算系统的欢迎. 本节将从离线划分和在线划分 2 类不同的划分方法介绍点划分的相关研究，并在最后对不同算法的优缺点进行了分析比较.

2.1 离线划分

KL 算法是 Kernighan 和 Lin[19] 提出的一种二部图划分算法. 算法的主要思想是为图划分定义一个增益函数，并通过交换顶点的方式找到最大化增益的方案. 具体来说，KL 算法先将图中的顶点随机地划分成 2 个部分，然后迭代地对图划分结果进行优化直到收敛. 在每次迭代中，KL 算法依次交换具有最大收益的顶点对，直到所有顶点完成 1 轮移动. 虽然每一轮迭代优化都采用贪心，易出现陷入局部最优解的情况，但 KL 算法允许增益为负的移动，所以仍然可能得到图划分的全局最优解. KL 算法的每次迭代的算法复杂度为 $O(|V|^3)$. Fiduccia 和 Mattheyses[20] 提出了 FM 算法改进了 KL 算法的搜索效率. 与 KL 算法类似，FM 同样定义了顶点移动的收益并迭代地对图划分进行优化. 然而不同于 KL 算法在每次迭代中对顶点进行交换，FM 通过选择移动收益最大的顶点到另一分区的策略，将 KL 算法单次迭代的复杂度由 $O(|V|^3)$ 降为 $O(|E|)$.

尽管降低了 KL 算法的算法复杂度，但 FM 算法的搜索空间仍然相当大，由文献 [47−48] 提出的多级图划分（multilevel graph partitioning, MGP）算法，由于其在时间开销和划分质量上的优越性，逐渐被研究人员广泛使用. Karypis 等人[5] 基于 KL 算法提出了 METIS 算法. 其主要特点是通过多轮的粗化操作对图中的顶点进行合并以降低图的规模，并在规模较小的图上运行 KL/FM 算法对图划分进行优化. 在粗化阶段，METIS 将联系紧密的顶点进行合并. 由于其与细化阶段最小化边割的目标一致，因此可以获得较好的划分质量. 此外 METIS 缩小了图的规模，降低了细化阶段的搜索空间规模，因此可以显著降低图划分的时间开销.

Wang 等人[28] 基于层次划分和标签传播的思想，为解决 10 亿级顶点规模的图划分问题提出 MLP 算法. 标签传播算法是一种经典的社区挖掘算法，其主要步骤为：1）每个顶点首先初始化自己的标签为顶点标签；2）通过迭代的方式更新顶点标签，在每次迭代中，顶点将自己的标签更新为邻居顶点标签中的最小值；3）顶点标签收敛，停止迭代. 此时具有相同标签的顶点将被划分到同一分区. 同时，MLP 基于 METIS 类似的层次划分框架，通过多轮标签算法对输入图进行粗化并在每轮粗化过程中将具有相同标签的顶点合并为 1 个超点. Wang 等人[22] 将 MLP 算法实现在 Trinity 中. 受益于 Trinity 高效的图探索和消息传递机制，MLP 算法易于实现并且具有较高的划分效率.

2.2 在线划分

流处理可以在有限的时间内快速地处理海量数据. 在大规模图数据处理中，在线划分可以快速地对图进行划分. 使用哈希函数随机划分顶点是最简单的在线划分策略，也是分布式图计算系统中最经常被使用的点划分算法. 随机哈希划分算法无需依赖拓扑结构等任何先验信息，因此是一种适用广泛的图划分策略，可以处理任意规模的图数据. 同时由于哈希过程中顶点之间相互独立，因此可以并行计算划分过程，进一步加快划分效率.

Stanton 等人[23] 首先将图划分问题建模于在线划分的场景，提出了 LDG (linear deterministic greedy) 算法. LDG 算法在划分过程中利用了历史信息对顶点进行分配. 具体而言，对于某个分区 P_i 以及顶点 v 的邻居列表 $N(v)$，顶点 v 所在的分区 $P(v)$ 由以下目标函数计算得出：

$$\arg\max_{i\in[1,k]}\left(|P_i\cap N(v)|\left(1-\frac{|P_i|}{C}\right)\right), \tag{1}$$

其中，$|P_i\cap N(v)|$ 表示顶点在每个分区的邻居个数，C 表示分区最大容量，负载越倾斜的顶点其在式 (1) 中具有的权重越低. LDG 总是将顶点 v 划分到其邻居最多的分区，并惩罚负载倾斜的分区从而达到最小化边割和负载平衡的目的. 相比于随机划分，LDG 算法可以获得更好的划分质量. 此外，由于要计算分区

容量 C, 顶点的总个数需要事先已知, 因此 LDG 算法不适合处理顶点个数未知的场景.

同时期提出的 Fennel 算法[24] 采用了类似的贪婪策略进行图划分. Fennel 算法将在线划分的目标建模成模块度(modularity)最大化问题并同时考虑最小化边割数以及最大化负载平衡 2 个优化目标. Fennel 算法的核心思想在于尽可能地将邻居顶点划分到同一分区并将非邻居顶点划分到不同分区. 顶点 v 所在的分区 $P(v)$ 主要通过下述优化函数计算得出:

$$\arg\max_{i\in[1,k]}\left(|P_i\cap N(v)|-\alpha\gamma|P_i|^{\gamma-1}\right), \tag{2}$$

其中, α 和 γ 是控制负载平衡的超参数. 式(2)中的 2 项分别对应不同的优化目标. 前者 $|P_i\cap N(v)|$ 的目的在于最小化切边数, 后者 $\alpha\gamma|P_i|^{\gamma-1}$ 则是为了控制负载均衡. 与 LDG 类似, Fennel 在划分过程中同样维护着划分的历史信息以辅助划分.

为了进一步降低图划分的边割数, Nishimura 等人[25] 利用重划分策略进一步对 LDG 和 Fennel 的划分结果进行优化并提出了 re-LDG 和 re-Fennel 算法. LDG 和 Fennel 通过计算 $|P_i\cap N(v)|$ 得到当前时刻顶点 v 在每个分区中的邻居个数, 这意味着在划分开始时, 多数顶点 $|P_i\cap N(v)|$ 将为 0, 而在接近划分结束的状态时, $|P_i\cap N(v)|$ 才能反映顶点在每个分区最终的邻居数量, 因此其算法的优化能力有限. 重划分策略将在线划分转换成迭代过程. 在每次重划分过程中, $|P_i\cap N(v)|$ 记录来自上一次读取的历史信息, 这也意味着图划分可以利用当前流中尚未看到的顶点分配信息. 尽管重划分策略并不能保证结果的收敛性, 但是经过多次重划分后, re-LDG 和 re-Fennel 算法可以取得与 METIS 算法近似甚至更好的图划分结果.

此外, 大部分在线流划分算法假设顶点及其邻接表的信息以流的方式读入并实时划分. Leopard[30] 和 IOGP[35] 假设图是以边的形式依次读入的. 这类算法的主要特点是将读取边的过程看作顶点邻接表更新的过程, 并随着邻接表的更新调整顶点的分区. Leopard 和 IOGP 都同样利用 Fennel 的优化目标函数(如式 (2)) 对不同的分区进行打分, 并将顶点重分配到得分最大的分区. 由于在图划分时无法获取顶点全部的邻居信息, 这类算法在划分质量上表现较差, 因此应用并不广泛.

2.3 小 结

总体而言, 离线划分策略往往能够得到更优的划分结果, 但同时会带来较大的资源开销; 与此同时在线划分策略能够更好地适用于大规模图划分任务, 在牺牲一定分区质量的情况下较快速地完成图分区任务. 表 1 具体展示点划分策略下的各个算法使用的具体图格式、优化目标及相应的优化策略.

Table 1 Graph Partitioning Techniques Under Vertex-Partitioning Strategy

表 1 点划分策略下的图划分技术

方法	算法	图格式	优化目标	优化策略
离线划分	KL[19]	邻接列表	切边数	顶点交换
	FM[20]	邻接列表	切边数	顶点移动
	METIS[5]	CSR	切边数/连通量	层次划分
	HMETIS[21]	CSR	切边数/连通量	层次划分
	Kahypar[27]	CSR	切边数/连通量	层次划分
	SHP[34]	CSR	概率扇出	启发式优化
	HYPE[33]	CSR	切边数	启发式优化
在线划分	LDG[23]	邻接列表	切边数	启发式优化
	Fennel[24]	邻接列表	模块度	启发式优化
	re-LDG[25]	邻接列表	切边数	多轮迭代
	re-Fennel[25]	邻接列表	切边数	多轮迭代
	Leopard[30]	SNAP	切边数	启发式优化
	IOGP[35]	SNAP	切边数	启发式优化
	Min-Max[29]	邻接列表	切边数	启发式优化

3 边划分策略

边划分策略将图划分成多个子图, 并且每个子图包含互不相交的边集. 由于在现实世界中, 顶点的度数基本存在着幂律分布, 高度数的顶点会产生更高的通信和计算开销, 因此点划分往往在幂律图的计算上表现不佳. 相反, 以边作为划分的基本单位可以更好地保证负载均衡, 从而提高分布式图计算的效率. 尽管在边划分策略的研究工作中, 仍然存在 KL/FM 算法类似的局部搜索算法 LS-G 和 LS-F 等, 但是边划分的研究中所面向的输入图的规模已经超过了百万顶点. 因此, 在边划分中研究者更关注于划分的效率, 这类方法包括经典的离线边划分算法 NE[36] 以及图流划分算法 Greedy[3], DBH[32], HDRF[31], Grid[26] 等. 下面将逐一进行介绍.

3.1 离线划分

Zhang 等人[36] 基于图的邻接性特征提出了邻居扩展(neighborhood expansion, NE)算法, 是目前使用最为普遍的离线边划分算法. NE 算法通过 p 轮的迭代生成图的划分结果 $p=|P|$. 在第 i 次迭代过程中, NE 算法选择图中尚未分配的边加入到 p_i 中, 直到分

区中边的个数达到分区容量$|E(P_i)|>\alpha m/p$，其中m和p分别是图中边的数目和分区数目，α为平衡系数. NE算法采用启发性的策略将边加入到当前分区中. 具体来说，NE算法共包含2条启发性策略：1）选择加入后顶点副本率增量最小的点，将该点加入$V(P_i)$，并将该点的所有边加入到$E(P_i)$；2）如果边的2个顶点均包含在$V(P_i)$中，该边可以加入到当前分区$E(P_i)$中，顶点副本率保持不变. 在这种启发性策略的保证下，NE算法所产生分区中能够保留图的局部性特征，从而生成高质量的图分区. 对于幂律图而言，使用NE算法进行图划分可以得到更低的副本率. 虽然NE是一种离线划分算法，其仍然在1 min内处理百万级别的图划分问题. 针对更大规模图的划分问题，Hanai等人[37]设计实现NE算法的分布式版本——Distributed NE算法，通过分布式架构对分区进行扩展，可以支撑10亿级别顶点的图划分问题.

借鉴KL/FM算法的局部搜索思想，Guo等人[43]提出了基于边划分的局部搜索算法LS-G和LS-F. LS-G和LS-F基于其他图划分算法得到初始的划分结果，并通过对分区中的某些边进行移动来进一步地提升图划分的质量. 由于大规模图数据边集的规模往往远大于点集的规模，边切分策略产生的搜索空间通常较大. 为了减少搜索空间的规模，Guo等人[43]提出了可移动边（adjustable edges）和块（blocks）的概念. 对于任何边分区P_i中的边$e=(u,v)$，如果存在另一个分区$P_j(i\neq j)$满足$(u,v)\in V(P_j)$，则称e是一条从P_i到P_j的可移动边. 对于分区内所有可移动边构成的集合$\{E_i^*, G(E_i)-E_i^*\}$，表示分区内不包含可移动边构成的子图，其中的每一条联通分量被视作块. 对于每一条P_i到P_j的可移动边，从P_i移动到P_j不会改变图划分的副本数；然而对于每一个块，由于顶点可能在其他分区存在副本，将块移动到这些分区可以减少副本数，从而提升图划分的质量. LS-G算法采用贪婪的策略找到最大收益的分区并对块进行移动. LS-F则采用最大流模型（max flow）找到独立的块集并同时对其进行移动. 由于LS-F算法在对分区进行搜索时同时考虑多个块进行移动，相比于LS-G算法，LS-F算法可以到达更多的搜索空间并得到更好的优化结果.

3.2 在线划分

类似于点划分，在边划分中使用哈希函数对边进行划分是最简单的在线划分策略. PowerGraph[3]提出了Greedy算法，通过检查边的端点在分区上的分布对边进行划分. 具体来说：1）当2个端点在当前划分中均已被分配，优先将边分配到负载最小的端点所在分区；2）若分区的并集为空或者仅有1个端点已被分配，将边分配到分区交集中负载最小的分区；3）若端点均未被分配，选择负载最小的分区对边进行划分. 策略的核心思想是根据历史的划分信息最小化顶点副本率，同时最大化负载均衡. Greedy算法可以生成较好的划分结果，然而由于算法需要维护顶点的划分结果，在分布式环境下具有较高的通信开销，因此划分效率低下. 同时Greedy算法极其依赖边的顺序，在划分中容易出现极端情况.

考虑到在幂律图中通常存在由低度数顶点构成的社区结构，同时高度数顶点连接了这些社区结构. 优化划分高度数顶点的边，以保留低度数顶点构成的社区不被划分可以获得更好的划分质量. DBH[32]（degree based hashing）提出一种对每条边的2个顶点度数较小的顶点进行哈希，从而得到边所在的分区的方法. 具体来说，对于边$e=(u,v)$和哈希函数$h(\cdot)$，若$degree(u)<degree(v)$，边e将会划分到分区$h(u)$，否则边e将会被划分到分区$h(v)$. DBH算法在边划分时保留了低度数顶点的局部性特征，在图中顶点度数存在倾斜时具有良好的划分质量.

HDRF[31]算法是另一种利用图的度数特征对图划分算法. 和DBH算法类似，HDRF算法的主要思想是针对高度数顶点的划分进行优化并保留低度数顶点的局部性特征. 由于在对边进行流式读取的过程中顶点的度数未知，HDRF利用顶点的局部度数（partial degree，即顶点目前已知的邻居个数）进行图化分. 在决定边$e=(u,v)$所在的分区时，端点的度数通过

$$\theta(u)=1-\theta(v)=\frac{degree(u)}{degree(u)+degree(v)}$$

进行归一化，并通过下述目标函数找到边将要划分的分区：

$$\arg\max\left(g(u,P_i)+g(v,P_j)+\lambda\left(1-\left|\frac{e(P_i^t)}{C}\right|\right)\right), \quad (3)$$

$$\text{where}, g(v,P_i)=(1+(1-\theta(v)))\times I(P_i).$$

HDRF倾向于将边划分到端点中度数较低的顶点所在的分区，因为低度数顶点计算得到的$g(v,P_i)$值更高. HDRF优化式（3）中的度数信息为局部度数，即读入当前边时顶点的度数. 由于此时尚未读取完整的图结构信息，此度数小于图中顶点的真实度数. 同时HDRF通过λ平衡负载和通信开销. 当$\lambda<1$时，HDRF生成的图划分结果和Greedy算法基本类似. 然而$\lambda>1$，HDRF将会取得更平衡的划分结果，以避免在特定情况下类似Greedy算法会将所有边划分到同一分区的结果.

Grid[26]算法是另一种基于哈希的图划分算法. 其主要将分区 ID 映射到一个 2 维的网格中. 每个顶点通过哈希函数$h(\cdot)$映射到网格中, 并且$h(\cdot)$所在的行和列中的所有分区构成顶点的约束集$S(\cdot)$. 在分配边$e=(u,v)$的分区时, 2 个端点分别被哈希到对应的约束集$S(u)$和$S(v)$中, 接着边 e 将会被分配到$S(u)\cap S(v)$中负载最小的分区. Grid 算法限制了顶点可以分配的分区, 可以保证图分区中顶点副本率不超过$2\sqrt{n}-1$. 但是 Grid 要求图分区的数目为n^2(n 为任意自然数). Grid 算法的另一个变种是 PDS[26]算法. PDS 通过完美差集(perfect difference sets)计算得到顶点的约束集. PDS 要求图分区的数目为p^2+p+1(p 为任意素数).

3.3 小 结

相比于点划分算法, 边划分可以在分布式图计算中取得更好的负载平衡, 策略总结如表 2 所示. 在离线边划分和流式图边划分算法的比较中, 二者的优缺点与离线点划分算法和在线点划分算法一致. 与 LDG 以及 Fennel 类似, Greedy 算法和 HDRF 等流式图划分算法使用了启发性策略, 同时 DBH 算法和 HDRF 算法考虑到了幂律图的度数特征, 进一步提升了图划分的质量.

Table 2 Graph Partitioning Techniques Under Edge-Partitioning Strategy

表 2 边划分策略下的图划分技术

方法	算法	输入	优化目标	优化策略
离线划分	NE[36]	SNAP	副本率	启发式策略
	Distributed NE[37]	SNAP	副本率	启发式策略
	LS-G/LS-F[43]	SNAP	副本率	局部搜索
在线划分	Greedy[3]	SNAP	副本率	启发式策略
	DBH[32]	SNAP	副本率	基于顶点度数
	HDRF[31]	SNAP	副本率	基于顶点度数
	Gird[26]	SNAP	副本率	2 维哈希
	PDS[26]	SNAP	副本率	完美差集

4 混合划分

从图的划分方式来看, 点划分将相同顶点的边划分到同一分区, 保留了顶点的局部性特征, 降低了通信开销; 而边划分将同一顶点的不同边划分到不同分区, 将计算负载分散到不同的计算节点中, 提高了负载平衡. 从图的处理方式来看, 离线划分利用全局信息对图进行划分, 可以取得相对较好的划分质量; 而在线划分实现了对图的实时处理, 具有较好的处理效率. 针对不同的应用需要以及图中顶点拓扑结构的特点, 采用结合不同划分策略的混合模式处理图数据, 往往能够更加契合相应的实际问题, 得到更加优秀的划分结果.

Hybird Ginger 算法是 Powerlyra[39]提出的一种混合图划分算法, 基于顶点的度数采用不同的图划分策略. 根据用户设定的阈值, Hybrid Ginger 算法将图中的顶点分为高度数顶点集合和低度数顶点集合. Hybrid 使用 High Cut 和 Low Cut 这 2 种不同的算法分别划分高度数顶点集和低度数顶点集. High Cut 是一种边划分算法, 使用哈希函数将边分散到不同的计算节点中, 达到负载平衡的目标; Low Cut 是一种点划分算法, 类似于 Fennel 算法的策略贪婪地选择顶点的分区并将顶点的边划分到同一分区. TopoX[38]提出的 TR 算法是另一种基于顶点度数的混合划分算法. TR 算法使用 Fission 和 Fusion 策略分别划分高度数顶点和低度数顶点. 其中 Fission 策略采用和 Hybrid Ginger 中 High Cut 相同的策略, 是一种边划分算法. Fusion 策略将图中相邻的顶点聚合成一个超点, 并将超点内部的顶点和边划分到相同分区以减少通信开销, 是一种点划分算法. 通过对顶点采用不同的划分策略, Hybrid Ginger 和 TopoX 均对图划分中降低通信开销和提高负载平衡 2 个重要的目标进行了优化.

在线划分算法大多依赖于历史的划分结果决定当前顶点或边的分区, 但由于在策略开始时缺乏历史信息, 导致划分质量较低且会影响后续的划分. 为了解决上述问题, OSPA[45]将边集分为 2 个不同的部分, 首先使用离线划分算法如 NE 算法对其中一部分进行划分, 得到一个较好的历史划分, 随后使用流式策略进行划分. OSPA 解决了在线划分算法对图流的顺序极为敏感的问题.

HEP[44]算法利用顶点的度数特征, 对高度数和低度数顶点集使用不同的图划分算法进行边的分配. 对于低度数顶点, HEP 算法扩展了 NE 算法并提出 NE++算法, 采用离线算法降低低度数顶点的副本率. 对于高度数顶点, HEP 算法使用 HDRF 算法对边进行在线划分. 考虑到低度数顶点的边较为稀疏, NE++算法使用 CSR 格式存储低度数顶点. 同时 NE++算法采用惰性边删除(lazy edge removal)的策略减少无向图的图划分开销. HEP 算法使用离线划分算法对低度数顶点进行划分, 降低了通信开销; 同时使用在线算法对高度数顶点进行划分, 降低了图划分的内存压力.

5 基于机器学习的方法

图划分将图中的顶点或边按照一定的策略划分到不同的分区，其本质是一类离散组合优化问题. 最小化边割或点割是图划分问题的优化目标，同时目标解需要满足分区平衡的约束. 基于机器学习的图划分算法目前仍是一个较新的领域，相关研究主要围绕图结构特征的提取以及图分区的建模问题.

GAP[41] 提出了一种通用的近似图分区求解框架，采用深度学习方法得到图分区. 具体而言，对于点划分问题，GAP 定义了一个可微分损失函数，通过训练一个神经网络生成具有最小边割的平衡分区并使用后向传播策略对网络的参数进行优化. GAP 利用图嵌入技术如 GCN 和 GraphSAGE 学习并表示图结构，使得其可以自适应不同结构的图数据. 同时，GAP 训练得到的模型可以对训练过程中不可见的图进行有效地划分，具有一定的泛化能力. 实验证明 GAP 在各种真实数据集和生成数据集上都有较好的表现，可以得到和 HMETIS 近似的划分质量，并且相比 HMETIS 其运行时间有 10~100 倍的性能提升.

GD[42] 算法将二分图划分问题建模成一个优化问题并采用梯度下降方法对问题进行求解. GD 通过将二分图划分问题转化一个 0-1 整形规划问题. 通过应用随机梯度下降递归地对解空间进行优化，GD 生成解向量并得到最终的划分结果. 同时算法在优化的过程中会随机地加入噪声，以避免陷入鞍点.

ADP[40] 认为需要根据图算法的特点选择不同的图划分策略. 对于某种特定的图算法，ADP 抽取分区特征（如顶点度数等），将图划分建模成一个多项式回归问题，学习得到其对应的代价模型，并通过代价模型预测图算法在分区上的计算开销和通信开销. 基于代价模型，ADP 拓展了点划分策略和边划分策略并提出了 ParE2H 和 ParV2H 算法能够根据代价模型调整分区的平衡并使用边 (点) 划分对图分区进一步进行优化.

Gatti 等人[46] 提出了一种基于强化学习的图划分模型. 该模型借鉴了层次划分的思想，在面对多分区任务时采用递归 2 分区的策略. 模型的奖励函数同时考虑了分区的平衡约束与切边数，保证其能够生成较高质量的分区结果. 实验证明，该模型在各类真实数据集上能够得到与 METIS 近似的划分质量，为帮助机器学习在图划分任务上的应用提供了新的思路.

6 超图划分

虽然图可以针对相互作用或者依赖关系进行建模，但是许多现实世界的问题涉及对象之间更复杂的关系. 例如对于超大规模集成电路设计而言，由于普通图模型无法简洁地表示 2 个以上对象之间的关系. 超图作为图的推广，其中每条超边都可以连接任意数量的顶点. 因此，与仅限于二元关系的图不同，超图可用于对更复杂的依赖关系和交互进行建模.

超图划分问题是图划分问题的推广，目标是将超图的顶点集划分到 k 个具有约束的分区，从而最小化切边数或者连通量. 然而，由于任意大小的超边而增加的灵活性使得超图分区在实践中比图分区更困难，因此超图划分算法也被认为在实施和运行时间方面更复杂. 由于应用众多，超图划分的相关研究已经引起了学术界和工业界的广泛关注.

HMETIS[21] 提出了一种针对超图的多层划分算法. 具体来说，HMETIS 首先通过多轮粗化操作对超图的顶点或者超边进行聚合以降低超图的规模，然后在规模较小的超图上使用基于 KL/FM 等启发式算法进行细化移动. 通过再粗化将超图规模进行缩小，从而减少 FM 等策略的解空间. 通过多轮粗化-细化迭代，最终能够以较低的开销获得较为优秀的划分结果.

Schlag 等人[27] 借鉴 HMETIS 的多级划分方法，提出了一种名为 KaHyPar 的超图划分算法. KaHyPar 同样采用多层划分的策略流程，通过对架构的更新与对各阶段策略的优化，其在保证超图划分质量的基础上实现比 HMETIS 更快的速度.

由于数据规模的原因，超图中同样出现了与普通图相对应的流式划分策略. Alistarh 等人[29] 提出了一种名为 Min-Max 的流式超图划分算法. Min-Max 算法每次会读取某一个顶点以及其相连的超边信息，在将该顶点分配分区时执行 Min, Max 这 2 个步骤完成分区动作. 策略首先会检查当前所有分区的负载情况，从中找出小于负载限制的分区，随后在筛选出的分区中，找到分区超边信息与当前顶点包含超边信息交集最大的分区，将顶点进行划分.

Mayer 等人[33] 根据在普通图中所提出的 NE 算法，针对超图特殊的结构进行设计提出了 HYPE 算法. HYPE 算法所采用的本质思想与 NE 算法相同，主要根据顶点的邻居信息将其分到关联密切的分区.

实验证明, 在超图规模较大的情况下, HYPE 能够较快地完成分区任务, 且比 Min-Max 等流分区算法有更加优秀的分区结果.

SHP[34] 提出了一种基于哈希计算的超图划分策略. 该策略将超图建模成二部图, 主要解决在面对真实社交场景时的查询请求的频繁扇出问题. 该策略设计了一个概率扇出的计算公式, 使其能够摆脱原始优化目标因局部最优导致无法有效优化的问题. 实验证明, 在面对超大规模超图的分区任务时, 该方法能够在较短的时间内实现较优的质量分区.

7 其他方法

谱聚类[49] 是一种经典的图划分策略, 通过聚类的方式将顶点划分到不同的分区. 谱聚类的优化目标是最小化分区间的边权, 使得相同分区内边权尽可能大. 基于谱划分的图划分算法是一类 NP 难问题. 为了降低计算复杂度, Spielman 等人[50] 通过近似算法将基于谱聚类的图划分算法的复杂度降低到了 $O(|E|/\gamma^2)(\gamma \in (0,1))$. 在此之上, Orecchia 等人[51] 采用原始对偶半定规划求解谱聚类问题, 将图划分的复杂度降低至 $O(|E|/\gamma)$.

和谱聚类类似, 标签传播[52] 也是一种经典的聚类方法, 其优化目标是最大化同一类簇内部的权重, 并最小化类簇间的权重. 标签传播有着和图划分类似的优化目标, 因此有一些研究者使用基于标签传播的策略对图进行划分. PuLP 算法[52] 使用标签传播将图划分为不同的社区, 每个社区内的顶点组成分区. XTRAPULP 算法[53] 是 PuLP 算法的一种分布式实现. MLP 算法[28] 使用多次标签传播策略迭代地对输入图进行粗化, 并使用其他图划分算法对图分区进行优化以产生更好的图分区.

另外, 由于图划分的复杂度较高, 为了提高图划分的效率, 一些学者研究了分布式和并行图划分方法, 如本文介绍过的 METIS[5] 的并行版本 ParMETIS[54] 以及 XTRAPULP[53] 等. 与此同时, GPU 是一类新的硬件架构, 比 CPU 拥有更多的计算单元, 可以为图划分问题提供更好的计算加速[55]. 高吞吐量同样是 GPU 在处理大规模图数据时的优势, 能够满足计算任务频繁调用数据的需求. 除此之外, GPU 在矩阵运算上的独特优势同样能够对图计算任务进行加速, 例如在进行图谱矩阵运算时, GPU 往往能够成倍地提升计算效率. Kim 等人[56] 充分利用 GPU 多核心的特性实现了一种新的图划分算法 Thanos.

8 未来研究方向和展望

尽管学术界和工业界在图划分领域取得了诸多研究成果, 但由于图数据应用的复杂性, 目前仍存在诸多挑战:

1)动态图的划分问题. 当前大多应用在分布式图计算系统中的图划分方法主要是处理静态的图数据, 然而现实世界的图数据往往具有高频变化的特性, 针对静态图的策略无法处理. 例如在电商等推荐系统中, 商品的动态更新非常频繁, 而基于图结构所构建的索引及其他应用的重构成本较高, 如何解决以图顶点为表示的商品更新一直都没有较好的处理方案. 目前有许多流式图划分算法研究如何针对时序图进行划分, 如利用快照技术将动态图转化为静态图[57-58], 然而其划分质量仍有待提高. 如何针对时序图数据设计有效的图划分模型, 动态地对图分区进行维护, 对分布式图计算系统具有重要的研究意义.

2)超图的划分问题. 超图作为一种新的图结构, 可以表达现实世界中复杂的多元关系, 应用愈加广泛. 例如在电路设计领域, 多个器件以及其连接关系通常被建模为超图处理, 如何合理地将器件分配规划到不同的电路板中是一个非常经典的超图划分问题. 但由于其复杂的多源关系以及数据规模的不断提高, 如何高效划分超大规模超图, 以及设计面向超图的并行图计算模型仍是一个新兴的研究领域.

3)基于机器学习策略的扩展与可解释问题. 基于机器学习的图划分仍是一个活跃的研究领域. 由于图结构特殊的性质, 基于机器学习的策略无法训练一个通用的模型适用于任意规模的图数据. 除此之外, 基于启发式的策略往往有强解释性, 这正是机器学习策略所空缺的部分, 这也导致其在各类真实应用场景中难以部署落地. 有效结合机器学习策略是当前图划分策略未来研究的热点.

4)新型计算节点下的图划分问题. 随着 GPU 等硬件设备的不断更新迭代, 当前分布式计算系统往往具有高速计算能力. 虽然当前研究例如文献 [56] 设计了适用于 GPU 的图划分策略, 但仍缺乏高效利用高性能计算硬件的研究. 特别是在大型分布式内存机上实现良好的可扩展性和质量仍然是一个挑战, 但即使在共享内存机上, 对大量线程的可扩展似乎也很困难. 此外, 更困难的是将最先进算法的固有复杂性和不规则性与 GPU 或 SIMD 指令的限制相一致.

充分利用高性能计算硬件能力，是图划分策略未来研究的重要方向之一.

9 结束语

本文首先介绍了分布式图计算系统以及图划分技术的相关背景，并介绍了当前分布式图计算系统中计算模型与图划分技术的关联与分类体系. 随后分别从点划分策略、边划分策略、混合划分策略、基于深度学习的策略、超图划分策略及其他划分策略的角度出发，分别介绍了其在分布式图计算系统中的代表性方法，并总结了不同策略在面对不同数据场景、应用场景中的优势与不足. 最后本文还讨论了分布式图计算系统中图划分技术面临的挑战与未来的发展方向. 分布式图计算系统中的图划分技术已经有了较为丰富的理论体系与较多方法的落地. 但随着当前数据规模的不断提升，基于 AI 大模型爆发带来的新数据的存储与响应需求的增加，以及基于机器学习模型的研究不断深入，未来会有更多的新研究和成果出现. 希望本文能够帮助相关领域研究人员把握当前研究进展与发展现状，为新数据与需求挑战下的图划分研究奠定坚实有力的基础，提供新的研究方向.

作者贡献声明：尚俊霖、张振宇与屈稳稳负责论文主要内容的撰写和校验；王晓玲对文章的结构和内容提供指导意见.

参 考 文 献

[1] Malewicz G, Austern M H, Bik A J C, et al. Pregel: A system for large-scale graph processing[C]//Proc of the 36th ACM SIGMOD Int Conf on Management of Data. New York: ACM, 2010: 135−146

[2] Low Y, Gonzalez J, Kyrola A, et al. Distributed GraphLab: A framework for machine learning in the cloud[J]. Proceedings of the VLDB Endowment, 2012, 5(8): 716−727

[3] Gonzalez J E, Low Y, Gu Haijie, et al. PowerGraph: Distributed graph-parallel computation on natural graphs[C]//Proc of the 10th Symp on Operating Systems Design and Implementation. Berkeley, CA: USENIX Association, 2012: 17−30

[4] Gonzalez J E, Xin R S, Dave A, et al. GraphX: Graph processing in a distributed dataflow framework[C]// Proc of the 11th Symp on Operating Systems Design and Implementation. Berkeley, CA: USENIX Association, 2014: 599−613

[5] Karypis G, Kumar V. A fast and high quality multilevel scheme for partitioning irregular graphs[J]. SIAM Journal on Scientific Computing, 1998, 20(1): 359−392

[6] Ching A, Edunov S, Kabiljo M, et al. One trillion edges: Graph processing at Facebook-scale[J]. Proceedings of the VLDB Endowment, 2015, 8(12): 1804−1815

[7] Tian Yuanyuan, Balmin A, Corsten S A, et al. From “think like a vertex” to “think like a graph”[J]. Proceedings of the VLDB Endowment, 2013, 7(3): 193−204

[8] Yan Da, Cheng J, Lu Yi, et al. Blogel: A block-centric framework for distributed computation on real-world graphs[J]. Proceedings of the VLDB Endowment, 2014, 7(14): 1981−1992

[9] Khayyat Z, Awara K, Alonazi A, et al. Mizan: A system for dynamic load balancing in large-scale graph processing[C]// Proc of the 8th ACM European Conf on Computer Systems. New York: ACM, 2013: 169−182

[10] Yan Da, Huang Yuzhen, Liu Miao, et al. GraphD: Distributed vertex-centric graph processing beyond the memory limit[J]. IEEE Transactions on Parallel and Distributed Systems, 2017, 29(1): 99−114

[11] Kyrola A, Blelloch G, Guestrin C. GraphChi: Large-scale graph computation on just a PC[C]// Proc of the 10th Symp on Operating Systems Design and Implementation. Berkeley, CA: USENIX Association, 2012: 31−46

[12] Li Jinji, Zhang Yanfeng, Gong Shufeng, et al. Streamlined asynchronous graph processing framework[J]. Journal of Software, 2018, 29(3): 528−544(in Chinese)
(李金吉，张岩峰，巩树凤，等. 流式处理的异步图处理框架[J]. 软件学报，2018, 29(3)：528−544)

[13] Roy A, Mihailovic I, Zwaenepoel W. X-Stream: Edge-centric graph processing using streaming partitions[C]//Proc of the 24th ACM Symp on Operating Systems Principles. New York: ACM, 2013: 472−488

[14] Cheng Jiefeng, Liu Qin, Li Zhengou, et al. VENUS: Vertex-centric streamlined graph computation on a single PC[C]// Proc of the 31st Int Conf on Data Engineering. Piscataway, NJ: IEEE, 2015: 1131−1142

[15] Roy A, Bindschaedler L, Malicevic J, et al. Chaos: Scale-out graph processing from secondary storage[C]//Proc of the 25th Symp on Operating Systems Principles. New York: ACM, 2015: 410−424

[16] Gao Pin, Zhang Mingxing, Chen Kang, et al. High performance graph processing with locality oriented design[J]. IEEE Transactions on Computers, 2017, 66(7): 1261−1267

[17] Bouhenni S, Yahiaoui S, Nouali-Taboudjemat N, et al. Efficient parallel edge-centric approach for relaxed graph pattern matching[J]. Journal of Supercomputing, 2022, 78(2): 1642−1671

[18] Kang S, Hastings C, Eaton J, et al. cuGraph C++ primitives: Vertex/edge-centric building blocks for parallel graph computing [C]//Proc of the 37th IEEE Int Parallel and Distributed Processing Symp Workshops (IPDPSW). Piscataway, NJ: IEEE, 2023: 226−229

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

[20] Fiduccia C M, Mattheyses R M. A linear-time heuristic for improving network partitions [C]//Proc of the 19th Design Automation Conf.

Piscataway, NJ: IEEE, 1982: 175−181

[21] Karypis G, Kumar V. Multilevel k-way hypergraph partitioning [C]//Proc of the 36th Annual ACM/IEEE Design Automation Conf. New York: ACM, 1999: 343−348

[22] Shao Bin, Wang Haixun, Li Yaotao. Trinity: A distributed graph engine on a memory cloud [C]// Proc of the 39th ACM SIGMOD Int Conf on Management of Data. New York: ACM, 2013: 505−516

[23] Stanton I, Kliot G. Streaming graph partitioning for large distributed graphs[C]//Proc of the 18th ACM SIGKDD Int Conf on Knowledge Discovery and Data Mining. New York: ACM, 2012: 1222−1230

[24] Tsourakakis C, Gkantsidis C, Radunovic B, et al. Fennel: Streaming graph partitioning for massive scale graphs[C]// Proc of the 7th ACM Int Conf on Web Search and Data Mining. New York: ACM, 2014: 333−342

[25] Nishimura J, Ugander J. Restreaming graph partitioning: Simple versatile algorithms for advanced balancing[C]// Proc of the 19th ACM SIGKDD Int Conf on Knowledge Discovery and Data Mining. New York: ACM, 2013: 1106−1114

[26] Jain N, Liao Guangdeng, Willke T L. Graphbuilder: Scalable graph ETL framework[C]//Proc of the 1st Int Workshop on Graph Data Management Experiences and Systems. New York: ACM, 2013: 19−24

[27] Schlag S, Henne V, Heuer T, et al. K-way hypergraph partitioning via n-level recursive bisection[C]//Proc of the 18th Workshop on Algorithm Engineering and Experiments (ALENEX). Philadelphia, PA: SIAM, 2016: 53−67

[28] Wang Lu, Xiao Yanghua, Shao Bin, et al. How to partition a billion-node graph[C]// Proc of the 30th Int Conf on Data Engineering. Piscataway, NJ: IEEE, 2014: 568−579

[29] Alistarh D, Iglesias J, Vojnovic M. Streaming Min-Max hypergraph partitioning[C]//Proc of the 28th Int Conf on Neural Information Processing Systems-Volume 2. Cambridge, MA: MIT, 2015: 1900−1908

[30] Huang Jiewen, Abadi D J. Leopard: Lightweight edge-oriented partitioning and replication for dynamic graphs[J]. Proceedings of the VLDB Endowment, 2016, 9(7): 540−551

[31] Petroni F, Querzoni L, Daudjee K, et al. HDRF: Stream-based partitioning for power-law graphs[C] //Proc of the 24th ACM Int on Conf on Information and Knowledge Management. New York: ACM, 2015: 243−252

[32] Xie Cong, Yan Ling, Li Wujun, et al. Distributed power-law graph computing: Theoretical and empirical analysis[C]//Proc of the 27th Int Conf on Neural Information Processing Systems-Volume 1. Cambridge, MA: MIT, 2014: 1673−1681

[33] Mayer C, Mayer R, Bhowmik S, et al. HYPE: Massive hypergraph partitioning with neighborhood expansion[C]//Proc of the 6th IEEE Int Conf on Big Data (Big Data). Piscataway, NJ: IEEE, 2018: 458−467

[34] Kabiljo I, Karrer B, Pundir M, et al. Social Hash partitioner: A scalable distributed hypergraph partitioner[J]. Proceedings of the VLDB Endowment, 2017, 10(11): 1418−1429

[35] Dai Dong, Zhang Wei, Chen Yong. IOGP: An incremental online graph partitioning algorithm for distributed graph databases[C]// Proc of the 26th Int Symp on High-Performance Parallel and Distributed Computing. New York: ACM, 2017: 219−230

[36] Zhang Chenzi, Wei Fan, Liu Qin, et al. Graph edge partitioning via neighborhood heuristic[C]//Proc of the 23rd ACM SIGKDD Int Conf on Knowledge Discovery and Data Mining. New York: ACM, 2017: 605−614

[37] Hanai M, Suzumura T, Tan W J, et al. Distributed edge partitioning for trillion-edge graphs[J]. arXiv preprint, arXiv: 1908.05855, 2019

[38] Zhang Yiming, Wang Haonan, Jia M, et al. TopoX: Topology refactorization for minimizing network communication in graph computations[J]. IEEE/ACM Transactions on Networking, 2020, 28(6): 2768−2782

[39] Chen Rong, Shi Jiaxin, Chen Yanchen, et al. Powerlyra: Differentiated graph computation and partitioning on skewed graphs[J]. ACM Transactions on Parallel Computing, 2019, 5(3): 1−39

[40] Fan Wenfei, Jin Ruochun, Liu Muyang, et al. Application driven graph partitioning[C]// Proc of the 46th ACM SIGMOD Int Conf on Management of Data. New York: ACM, 2020: 1765−1779

[41] Nazi A, Hang W, Goldie A, et al. GAP: Generalizable approximate graph partitioning framework[J]. arXiv preprint, arXiv: 1903.00614, 2019

[42] Avdiukhin D, Pupyrev S, Yaroslavtsev G. Multi-dimensional balanced graph partitioning via projected gradient descent[J]. Proceedings of the VLDB Endowment, 2019, 12(8): 906−919

[43] Guo Zhenyu, Xiao Mingyu, Zhou Yi, et al. Enhancing balanced graph edge partition with effective local search[C] //Proc of the 35th AAAI Conf on Artificial Intelligence. Palo Alto, CA: AAAI, 2021: 12336−12343

[44] Mayer R, Jacobsen H A. Hybrid edge partitioner: Partitioning large power-law graphs under memory constraints[C]//Proc of the 47th Int Conf on Management of Data. New York: ACM, 2021: 1289−1302

[45] Ayall T, Duan Hanchong, Liu Changhong, et al. Taking heuristic based graph edge partitioning one step ahead via OffStream partitioning approach[C]//Proc of the 37th IEEE Int Conf on Data Engineering (ICDE). Piscataway, NJ: IEEE, 2021: 2081−2086

[46] Gatti A, Hu Zhixiong, Smidt T, et al. Graph partitioning and sparse matrix ordering using reinforcement learning and graph neural networks[J]. Journal of Machine Learning Research, 2022, 23(1): 13675−13702

[47] Bui T N, Jones C. A heuristic for reducing fill-in in sparse matrix factorization[R]. Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM), 1993

[48] Cheng C K, Wei Y C A. An improved two-way partitioning algorithm with stable performance[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 1991, 10(12): 1502−1511

[49] Shi Jianbo, Malik J. Normalized cuts and image segmentation[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2000, 22(8): 888−905

[50] Spielman D A, Teng Shanghua. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear

systems[C]//Proc of the 36th Annual ACM Symp on Theory of Computing. New York: ACM, 2004: 81−90

[51] Orecchia L, Vishnoi N K. Towards an SDP-based approach to spectral methods: A nearly-linear-time algorithm for graph partitioning and decomposition[C]//Proc of the 22nd Annual ACM-SIAM Symp on Discrete Algorithms. Philadelphia, PA: SIAM, 2011: 532−545

[52] Slota G M, Madduri K, Rajamanickam S. PuLP: Scalable multi-objective multi-constraint partitioning for small-world networks [C]//Proc of the 2nd IEEE Int Conf on Big Data (Big Data). Piscataway, NJ: IEEE, 2014: 481−490

[53] Slota G M, Rajamanickam S, Devine K, et al. Partitioning trillion-edge graphs in minutes[C]//Proc of the 31st IEEE Int Parallel and Distributed Processing Symp (IPDPS). Piscataway, NJ: IEEE, 2017: 646−655

[54] Karypis G, Kumar V. A parallel algorithm for multilevel graph partitioning and sparse matrix ordering[J]. Journal of Parallel and Distributed Computing, 1998, 48(1): 71−95

[55] Ye Chang, Li Yuchen, He Bingsheng, et al. Large-scale graph label propagation on GPUs[J/OL]. IEEE Transactions on Knowledge and Data Engineering, 2023[2023-12-01]. https://ieeexplore.ieee.org/abstract/document/10330123

[56] Kim D H, Nagi R, Chen Deming. Thanos: High-performance CPU-GPU based balanced graph partitioning using cross-decomposition [C]//Proc of the 25th Asia and South Pacific Design Automation Conf (ASP-DAC). Piscataway, NJ: IEEE, 2020: 91−96

[57] Massri M, Miklos Z, Raipin P, et al. Clock-G: A temporal graph management system with space-efficient storage technique[C]//Proc of the 38th IEEE Int Conf on Data Engineering (ICDE). Piscataway, NJ: IEEE, 2022: 2263−2276

[58] Li He, Liu Yanna, Yuan Hang, et al. Research on dynamic graph partitioning algorithms: A survey[J]. Journal of Software, 2023, 34(2): 539−564 (in Chinese)
(李贺, 刘延娜, 袁航, 等. 动态图划分算法研究综述[J]. 软件学报, 2023, 34(2): 539−564)

Shang Junlin, born in 2000. Master candidate. Student member of CCF. His main research interests include graph computing, graph data mining, and database system.

尚俊霖, 2000 年生. 硕士研究生. CCF 学生会员. 主要研究方向为图计算、图数据挖掘、数据库系统.

Zhang Zhenyu, born in 1995. PhD candidate. His main research interests include graph data processing and large-scale graph data system. (zyzhang0731@163.com)

张振宇, 1995 年生. 博士研究生. 主要研究方向为图数据处理、大规模图数据系统.

Qu Wenwen, born in 1995. PhD. His main research interests include database, graph computing and distributed graph systems. (wenwenqu@stu.ecnu.edu.cn)

屈稳稳, 1995 年生. 博士. 主要研究方向为数据库、图计算和分布式图系统.

Wang Xiaoling, born in 1975. PhD, professor. PhD supervisor. Senior member of the CCF. Her main research interests include knowledge graph, personalized recommendation and privacy protection. (xlwang@cs.ecnu.edu.cn)

王晓玲, 1975 年生. 博士, 教授, 博士生导师. CCF 高级会员. 主要研究方向为知识图谱、个性化推荐、隐私保护.