软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, 2023,34(2):539−564 [doi: 10.13328/j.cnki.jos.006705]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

动态图划分算法研究综述 *

李 贺[1], 刘延娜[1], 袁 航[2], 杨舒琪[1], 韵晋鹏[1], 乔少杰[3], 黄健斌[1], 崔江涛[1]

[1](西安电子科技大学 计算机科学与技术学院, 陕西 西安 710126)

[2](字节跳动, 浙江 杭州 310020)

[3](成都信息工程大学 软件工程学院, 四川 成都 610225)

通信作者: 李贺, E-mail: heli@xidian.edu.cn

摘 要: 图划分是大规模分布式图处理的首要工作, 对图应用的存储、查询、处理和挖掘起基础支撑作用. 随着图数据规模的不断扩大, 真实世界中的图表现出动态性. 如何对动态图进行划分, 已成为目前图划分研究的热点问题. 从不同动态图划分算法的关注点和特点出发, 系统性地介绍当前可用于解决动态图划分问题的各类算法, 包括流式图划分算法、增量式图划分算法和图重划分算法. 首先介绍图划分的 3 种不同的划分策略及问题定义、图的两种不同的动态性来源以及动态图划分问题; 然后介绍 3 种不同的流式图划分算法, 包括基于 Hash 的划分算法、基于邻居分布的划分算法以及基于流的优化划分算法; 其次介绍单元素增量式划分和批量增量式划分这两种不同的增量式图划分算法; 再次, 分别介绍针对图结构动态的重划分算法和针对图计算动态的重划分算法; 最后, 在对已有方法分析和比较的基础上, 总结目前动态图划分面临的主要挑战, 提出相应的研究问题.

关键词: 图划分; 动态图; 分布式图处理; 图算法

中图法分类号: TP301

中文引用格式: 李贺, 刘延娜, 袁航, 杨舒琪, 韵晋鹏, 乔少杰, 黄健斌, 崔江涛. 动态图划分算法研究综述. 软件学报, 2023, 34(2): 539–564. http://www.jos.org.cn/1000-9825/6705.htm

英文引用格式: Li H, Liu YN, Yuan H, Yang SQ, Yun JP, Qiao SJ, Huang JB, Cui JT. Research on Dynamic Graph Partitioning Algorithms: A Survey. Ruan Jian Xue Bao/Journal of Software, 2023, 34(2): 539–564 (in Chinese). http://www.jos.org.cn/1000-9825/6705.htm

Research on Dynamic Graph Partitioning Algorithms: A Survey

LI He[1], LIU Yan-Na[1], YUAN Hang[2], YANG Shu-Qi[1], YUN Jin-Peng[1], QIAO Shao-Jie[3], HUANG Jian-Bin[1], CUI Jiang-Tao[1]

[1](School of Computer Science and Technology, Xidian University, Xi'an 710126, China)

[2](ByteDance, Hangzhou 310020, China)

[3](School of Software Engineering, Chengdu University of Information Technology, Chengdu 610225, China)

Abstract: Graph partitioning is the primary work of large-scale distributed graph processing, which plays a fundamental role in storage, query, processing, and mining of graph applications. Since graph data in the real world are always dynamic, the research of dynamic graph partitioning is a hot topic. This study systematically introduces the current algorithms for dynamic graph partitioning, which including streaming graph partitioning algorithm, incremental graph partitioning algorithm, and graph repartitioning algorithm. Firstly, the study introduces three different partitioning strategies, two different dynamic sources of graph and dynamic graph partitioning problem. Then, three different streaming graph partitioning algorithms are introduced, including hash algorithm, neighbor distribution-based algorithm, and novel algorithm. Secondly, two different incremental graph partitioning algorithms, single element incremental graph partitioning, and

* 基金项目: 国家自然科学基金(61602354, 61876138); 四川省科技计划(2021JDJQ0021, 2022YFG0186); 成都市技术创新研发项目(2021-YF05-00491-SN); 成都市重大科技创新项目(2021-YF08-00156-GX); 成都市"揭榜挂帅"科技项目(2021-YF08-00156-GX); 中央高校基本科研业务费专项

收稿时间: 2021-12-11; 修改时间: 2022-02-22, 2022-05-09; 采用时间: 2022-05-26; jos 在线出版时间: 2022-07-22

batch incremental graph partitioning are introduced. Thirdly, the repartitioning algorithm for graph structure and the repartitioning algorithm for graph computation are introduced, respectively. Finally, based on the analysis and comparison of the existing methods, the main challenges of dynamic graph partitioning are summarized and the corresponding research problems are proposed.

Key words: graph partitioning; dynamic graphs; distributed graph processing; graph algorithms

近年来, 随着互联网信息技术的迅猛发展, 各行各业每时每刻都在产生着各种各样的数据. 图作为一种特殊的数据结构, 不但可以存储数据个体本身, 还可以存储数据个体之间复杂的关联关系, 吸引了学术界和工业界的广泛关注. 真实数据集的来源非常广泛, 除了一些网站[1−4]收集的数据集之外, 还有从社交网站、电商网站的 API 以及与公司合作获得的数据集. 除此之外, 一些工作[5−7]还提供了生成人工数据集的工具. 现实当中, 我们所面临的图数据通常规模庞大、结构复杂且呈动态增长态势. 例如: Google 公司存储的知识图谱(knowledge graph)数据已包含超过 570 亿个数据对象, 这些对象之间还存在着复杂的语义链接关系; Twitter 的每月活跃用户在 10 年内从 4.82 亿增长到 25 亿; 万维网中网页的数量在 5 年内从 400 亿增长到 550 亿. 在集中式的图存储和图处理无法满足需求的情况下, 大量的分布式图处理系统应运而生, 如 Pregel[8]、Giraph[9]、GraphLab[10]、Powergraph[11]、Powerlyra[12]、GraphX[13]、TopoX[14]、RGraph[15]等.

图划分是分布式图处理的基础工作, 其作用是将一个大规模图数据划分到一个集群中的不同机器上. 在分布式图处理系统中, 根据短板效应, 集群的处理速度取决于其中运行最慢的机器, 这就需要集群中各个机器的工作负载应该尽可能相同. 同时, 图处理任务需要沿着图的拓扑结构进行, 这使得一个机器中的某些顶点需要与其他机器中的邻居顶点通过传递消息进行通信. 由于网络带宽和网络延迟等因素, 集群中不同机器之间的通信代价和通信时间要远远高于单个机器内部的通信成本, 且不同机器之间的通信受网络性能的影响而具有不稳定性, 这就需要集群中不同机器之间的通信成本要尽可能地低. 因此, 分布式图计算的性能主要由运行最慢的机器和不同机器之间的通信成本决定. 所以, 图划分的质量对分布式图处理的性能有很大的影响, 其目标是在集群中各个机器负载平衡的基础上, 最小化不同机器之间的通信成本.

基于负载平衡和最小化通信成本这两个基础目标, 很多不同的图划分算法被提出. 最早的划分算法主要是针对静态图的, 如谱划分算法[16,17]、几何划分算法[18,19]、多级图划分算法[20−22]、流式图划分算法[23,24]、分布式图划分算法[25]等. 但是, 实际上, 现实中的图大多数都具有一定的动态性, 这给图划分问题提出了新的挑战. 因此, 近年来, 很多研究者提出了动态图划分算法, 如批处理增量式图划分算法[26]、流式动态图划分算法[27−34]、动态重划分算法 Hermes[35]、GraphH[36]、LogGP[37]等. 有一些综述文章对图划分算法进行了不同角度的总结和归纳, 但它们提到的算法大多都用于划分一个静态图. 文献[38,39]对目前主流的静态图划分算法进行了详细的分类和总结. 文献[40,41]则对流式图划分算法进行了总结. 文献[42]对不同的分布式图处理系统中的图划分算法进行了总结, 主要讨论了流式图划分算法, 但更侧重于不同框架的性能比较. 文献[43,44]虽然提到了动态图划分, 但并没有进行详细的分类总结. 本文将聚焦当今热门的动态图划分问题, 对现有的动态图划分算法进行分析、总结和归纳.

现有的动态图划分算法与静态图划分算法有着明显的区别: 一是动态图划分算法通常可以重用初始划分的信息, 从而减少动态划分的开销, 这也是动态图划分对比静态图划分的优势; 二是图的动态性是持续的, 所以动态图划分算法的开销不能过大. 因此, 利用静态图划分算法重头开始来处理动态图是不可行的. 目前, 可用于划分动态图的划分算法从不同的关注点和特点出发, 主要可分为流式动态图划分算法、增量式图划分算法和图重划分算法. 这样的分类依据主要是由于流式动态图划分算法的主要特点是能够实时地依次处理单个顶点或者边, 它能够以较高的算法效率处理不断变化的动态图. 但大部分的流式动态图划分算法过于追求低的划分时间和划分成本, 因而获得了较低的划分质量. 此外, 大部分流式图划分算法最初被提出是针对大规模图划分的. 增量式图划分算法与图重划分算法的主要区别在于: 增量式图划分算法提出了专注于处理动态变化的那部分图数据, 对动态变化部分提出了划分策略; 而图重划分算法则假设图发生动态变化已经导致分区发生了变化, 它更专注于通过动态调整不断优化分区, 从而获得高质量的划分结果.

图 1 给出了动态图划分算法的分类框架, 动态图划分算法主要分为流式动态图划分算法、增量式图划分

算法以及图重划分算法. 在流式动态图划分算法中, 根据不同算法在算法效率和划分质量上的平衡程度, 可分为基于哈希的划分算法、基于邻居分布的划分算法和基于流的优化划分算法. 增量式图划分算法根据专注于处理动态变化的规模大小, 可分为单元素增量式划分算法和批处理增量式划分算法. 图重划分算法根据针对图结构发生动态变化还是针对图计算发生动态变化来优化分区, 可分为针对结构动态的划分算法和针对计算动态的划分算法.

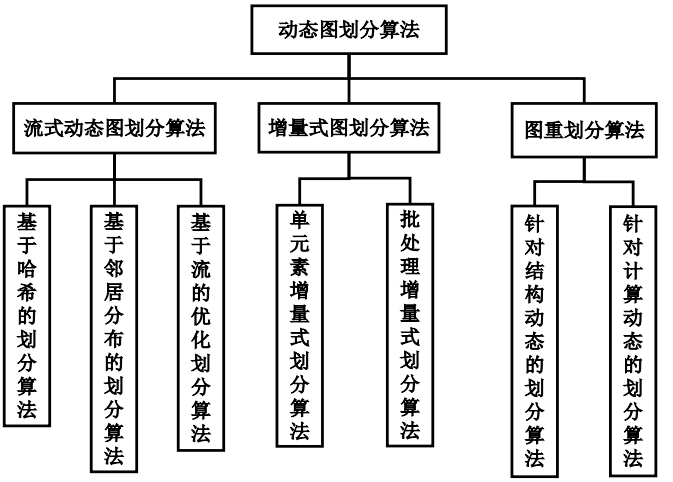


图 1 动态图划分算法分类

本文在对近年来动态图划分问题的研究进程进行跟踪分析的基础上, 对该问题的相关工作进行了深入的分析、总结和归纳. 第 1 节概述图划分的背景. 第 2 节介绍流式动态图划分算法. 第 3 节介绍增量式图划分算法. 第 4 节介绍图重划分算法. 第 5 节对动态图划分问题的研究进行总结和分析, 讨论其面临的挑战, 并展望未来的研究方向.

1 背景概述

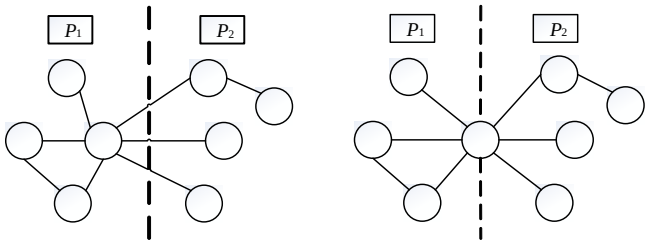
图划分是一个经典的 NP 难问题, 最初在文献[45–47]中被用于将电子电路的组件分配给电路板, 以最大程度地减少电路板之间的连接数. 随着图划分领域的发展, 图划分被广泛用于许多场景, 如分布式图存储、查询、计算等. 不同的场景对图划分的目标提出了不同的要求, 因此图的划分方式也逐渐多样化. 近年来, 随着应用场景中数据量的快速增长, 图划分的目标逐渐转移到了各种类型的动态图上. 本节将主要介绍不同的划分方式和图划分问题、传统的图划分算法、图的动态性来源以及动态图划分问题.

1.1 图的划分方式

一个图由顶点和边构成, 因此划分一个图的基本方式主要有顶点划分和边划分两种. 顾名思义, 顶点划分是将图中的顶点划分成多个分区, 若边的两个端点被划分到两个不同的分区, 则该边被切割, 因此, 顶点划分也被称为基于边割的图划分. 图 2(a)是顶点划分的一个例子, 分区 P_1 和 P_2 各包含 4 个顶点, 且有 3 条边被切割. 边划分则是将图中的边划分成多个分区, 若顶点的邻边被划分到多个分区, 则该顶点需要在对应的分区创建副本顶点, 即该顶点被切割, 因此, 边划分也被称为基于点割的图划分. 图 2(b)是边划分的一个例子, 分区 P_1 和 P_2 各包含 4 条边, 且有一个顶点被切割成两个副本.

边划分的提出是为了解决顶点划分的缺陷, 最初, 分布式图处理系统使用的是顶点划分, 例如 Pregel、Giraph、GraphLab 等. 在这些系统中, 被切割的边需要存储在其端点所在的两个分区中, 这导致每个顶点的计算负载都集中在一个分区内, 具有较强的局部性. 而在真实应用中, 图的度通常都呈幂律分布, 即大多数顶点只有相对较少的邻居, 而少数顶点有非常多的邻居, 例如社交网络中的明星和普通人, 一般来说, 明星有很多

人关注，而普通人有很少的人关注. 在幂律度分布下，顶点的度为 d 的概率为 $P(d)\propto d^{-\alpha}$. 其中，指数α是一个控制度分布的偏斜程度偏度的正整数. 较高的α意味着图具有较低的密度，且绝大多数顶点都是低度的. 随着α的降低，图的密度和高度顶点的数量增加. 大多数真实图的幂律常数通常大约是$\alpha\approx 2$. 倾斜的度分布意味着一小部分顶点与大部分边相邻，在顶点划分的环境下，度高的超级顶点会导致分布式图计算的计算负载不均衡、通信量不均衡和存储负载不均衡. 实际上，每个分区的计算负载、通信负载和存储负载都与边的数量线性相关，因为顶点的值都通过边传播，且以邻接列表的形式存储数据. 因此，边划分更加契合具有幂律分布的真实图的划分问题. 由此，一些研究者提出了基于边划分的分布式图处理系统，例如 PowerGraph、GraphX 等.



(a) 顶点划分 (b) 边划分

图 2 顶点划分与边划分

早期的像 Pregel、Giraph 和 GraphLab 的图并行系统通过切割边划分一个图以均匀地分配顶点，这些基于顶点划分的分布式图处理系统会遭受高度顶点所带来的不平衡的计算和通信负载. 由此，像 PowerGraph 和 GraphX 的图并行系统通过切割顶点以均匀地在多个机器上分配边，尽管边划分减轻了高度顶点带来的不平衡问题，但是基于边划分的系统对于低度顶点来说会招致较高的通信成本和过多的内存消耗. 因此，最近，Powerlyra 提出了混合切割策略来解决这个问题，区别对待高度顶点和低度顶点. 具体地，将高度顶点的边分配给所有机器来均匀分配计算负载，以及将低度顶点划分到同一台机器以减少不必要的通信. 混合切割的划分方式是顶点划分方式和边划分方式的结合. 图 3 是混合切割的一个例子，原图被划分为 3 个分区 P_1、P_2、P_3. 其中，顶点 1 是高度顶点，它的邻边由边划分算法划分到不同的分区中(如右边的中间部分)，而其余的顶点都是低度顶点，由顶点划分算法划分到不同的分区(如右边的上半部分)，二者的结合就是混合切割策略得到的划分(如右边的下半部分).

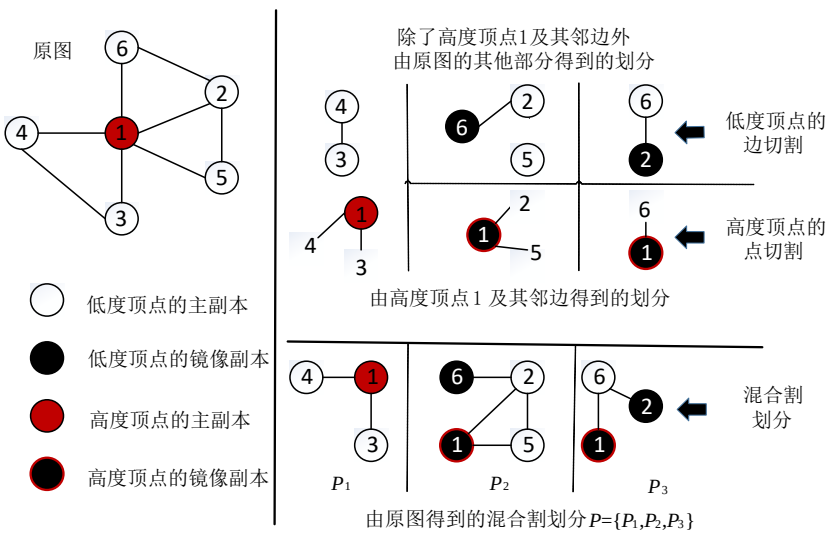


图 3 混合割划分

1.2 图划分问题定义

给定一个图 $G=(V,E)$, V 是图的顶点集合, E 是图的边集合, 对于任意子集 $E_s\subseteq E$, $V[E_s]$表示在 E_s 中所有顶点的集合. 同样地, 对于任意子集 $V_s\subseteq V$, $E[V_s]$表示两个端点都在 V_s 中的边的集合. 用 $N(v)$表示顶点 v 的邻居顶点的集合. 图 G 被划分成 k 个分区, 其中, k 是一个远小于$|V|$和$|E|$的正整数.

1.2.1 基于边割的图划分问题

顶点划分的划分对象是图中的顶点集合 V, 若图 G 被划分成 k 个分区 $P=\{P_1,P_2,\ldots,P_k\}$, 满足 $V=\bigcup_{i=1}^{k}P_i$ 且当 $i\neq j$ 时 $P_i\cap P_j=\varnothing$, 其中, P_i 是 P 的第 i 个分区, 那么我们称 P 是图 G 的一个顶点划分. 对于一个顶点的子集 $S\subseteq V$, 用 $E(S,V\backslash S)$表示顶点子集和 V 的其他顶点连接边的集合, 并且用$|E(S,V\backslash S)|$表示顶点子集和 V 的其他顶点之间连接边的数量. 在分布式图计算中, 不同分区的顶点之间需要传递消息, 而不同分区之间的消息传递成本较高, 因此需要最小化不同分区顶点之间边的数量. 此外, 还需要保证每个分区的负载尽可能相同, 即每个分区的顶点数几乎相同. $EC(P)$表示的是在图 G 的一个顶点划分 P 下的边切割, 因此顶点划分的问题定义如下:

$$\left.\begin{aligned}&\min\ EC(P)=\frac{1}{2}\sum_{i=1}^{k}|E(P_i,V\setminus P_i)|\\&\text{s.t.}\quad |P_i|\leqslant(1+\varepsilon)\frac{|V|}{k}\end{aligned}\right\}\tag{1}$$

其中, ε是满足 $0\leqslant\varepsilon\leqslant 1$ 的不平衡系数. 这里, $\varepsilon=0$ 表示每个分区具有相同顶点数量的最严格情况, 而$\varepsilon=1$ 表示每个分区的顶点数量可能高达平均顶点数量的两倍.

1.2.2 基于点割的图划分问题

边划分的划分对象是图中的边集合 E, 若图 G 被划分成 k 个分区 $P=\{P_1,P_2,\ldots,P_k\}$, 满足 $E=\bigcup_{i=1}^{k}P_i$ 且当 $i\neq j$ 时 $P_i\cap P_j=\varnothing$, 其中, P_i 是 P 的第 i 个分区, 那么我们称 P 是图 G 的一个边划分. 在边划分中, 每个顶点的邻边可能会被划分到不同的分区, 那么顶点 $v\in V$ 可能被切割, 即在多个分区拥有一个副本. 用 $R(v)$表示 v 所在分区集合, 那么$|R(v)|$就表示顶点 v 拥有的副本数量. 在分布式图计算中, 被切割的顶点有一个为主顶点, 其他为副本顶点. 主顶点与副本顶点之间需要同步消息, 而消息同步会使性能下降, 因此需要最小化顶点的副本数量. 此外, 还需要保证每个分区的负载尽可能相同, 即每个分区的边数几乎相同. $VC(P)$表示的是在图 G 的一个边划分 P 下的点切割, 因此边划分的问题定义如下:

$$\left.\begin{aligned}&\min\ VC(P)=\sum_{v\in V}|R(v)|-|V|\\&\text{s.t.}\quad |P_i|\leqslant(1+\varepsilon)\frac{|E|}{k}(1\leqslant i\leqslant k)\end{aligned}\right\}\tag{2}$$

与顶点划分类似, 其中, ε是满足 $0\leqslant\varepsilon\leqslant 1$ 的不平衡系数. 这里, $\varepsilon=0$ 表示每个分区具有相同边数量的最严格情况, 而$\varepsilon=1$ 表示每个分区的边数量可能高达平均边数量的两倍.

1.2.3 基于混合割的图划分问题

混合割划分对低度顶点和高度顶点的处理是不同的. 对于高度顶点, 混合割将高度顶点的邻边均匀划分到多个分区中, 这遵循点切割划分, 其主要目的是使得分区的负载尽可能地相等. 对于低度顶点, 混合割将低度顶点的邻边划分到同一分区, 这遵循边切割划分, 其主要目的是尽可能地减少不同分区间的通信. 在混合割中, 每个顶点通过哈希其 ID 分配给一个分区. 当一条边 e 和其中一个端点 s 被一起划分到一个分区, 而该条边的另外一个端点 t 被划分到另外一个分区时, 则称 s 在第 1 个分区中是主副本, t 在第 2 个分区中是主副本, s 和 t 在其他分区中被称为镜像副本.

1.3 静态图划分算法

针对静态图划分问题, 研究者们已经提出了大量的图划分算法, 主要可以分为基本图划分算法、多级图划分算法、分布式图划分算法以及其他类型图划分算法.

- 边切割

在基于边割的静态图划分算法中, 最经典的方法是 Metis[20−22], 它使用了多级划分的策略. 然而, Metis 是集中式的划分算法, 它无法满足超过百万个顶点的大图的划分需求. 除 Metis 之外, 还有 Scotch[48]、chaco[49]、文献[50]等图划分算法也使用了多级划分策略, 但它们同样在扩展性上有缺陷. 为了提高可扩展性, 出现了分布式的图划分算法, 如 ParMetis[51]、PT-Scotch[52]、JA-BE-JA[53]等. Metis 和 ParMetis 家族针对不同场景提出了很多算法, 这些算法及软件都可以在相关网站[54]中找到. 此外, 还有一些求精算法用来不断优化分区, 如 KL[55]、FM[56]、HS[57]、文献[58]中所提算法等. 文献[59]发现了目前静态图划分中存在的一些问题, 文中提出了多级标签传播算法 MLP. Spinner[60]也使用了标签传播算法对整个图进行划分. 文献[61]根据顶点度的大小, 将图分成低度顶点和高度顶点这两部分分别进行划分. 此外, 文献[62]引用了抗偏斜的图划分思想, 它提出了多标签图划分算法 MLGP.

- 点切割

在基于点割的图划分中, 有些算法借鉴了基于边割的图划分方法, 如 SBV[63]、JA-BE-JA-VC[64]等. 在基于点割的静态图划分算法中, NE[65]是目前划分效果最好的离线算法, 它通过邻居扩展的策略来依次生成每一个分区. 类似的基于邻居扩展的方法还有文献[66]中提出的算法. 文献[25]对 NE 进行了分布式的扩展. 另外, 还有一些基于点割的图划分算法运用了新颖的策略, 也有比较好的效果, 如 VESP[67]、SGVCut[68]等. Sheep[69]是一个分布式图划分器, 它先对顶点进行排序, 按顺序将输入图缩减为一个小的消除树, 在消除树上进行图划分, 最后再将树还原成图. 这个方法对顺序不敏感, 且易于扩展, 它在没有先验数据分布的情况下对大图进行分区. Dfep[70]也是一个分布式基于点割的图划分算法, 它引入货币的概念让分区购买边.

- 混合切割

PowerLyra[12]提出了混合切割的策略, 将低度顶点和高度顶点分开考虑. 它结合边切割和点切割, 其中, 点切割策略只用来切割高度顶点, 高度顶点由用户定义的阈值控制. TopoX[71]认为: 现有的算法都只是切割单条边或单个顶点, 实际上并没有利用图中的拓扑信息. 因此, 它提出了图的重构, 将低度顶点聚合成一个超级顶点, 将高度顶点切割成多个子顶点. 随后, 对重构后的图进行划分. 此外, Gemini[72]和 MDBGP[73]通过结合基于平衡度量的顶点和边负载来平衡混合工作负载. 文献[74]提出了一个应用驱动的混合划分策略.

此外, 文献[75]定义了分布式迭代计算环境下图划分的概念, 这与传统的通用图划分不同, 设计了一种基于定向边交换模型的分布式在线图划分算法(OnFlyP). 文献[76]总结了知识图谱划分算法, 包括基本、多级、流式、分布式和其他类型图划分算法. 文献[77,78]研究了超图划分问题. 以上所有的图划分算法都是用来处理静态图的, 而在真实世界中, 大部分图是动态变化的, 有些研究者已经提出了针对动态图的划分算法, 但是目前还没有文献对其进行很好的总结.

1.4 图的动态性来源及动态图划分问题

1.4.1 图的动态性来源

对于结构动态而言, 图 $G=(V,E)$中存在顶点和边的插入和删除. 在一段特定的时间内, 图 G 转变为图 $G'=(V',E')$, $V'=V\cup V_I-V_D$, $E'=E\cup E_I-E_D$, 其中, V_I 和 E_I 分别表示图中新插入的顶点和边的集合, V_D 和 E_D 分别表示图中被删除的顶点和边的集合. 图 G'被划分成 k 个分区 $P'=\{P'_1,P'_2,...,P'_k\}$, 因此, 动态图划分的目标是使 P' 在负载均衡的前提下最小化通信代价. 图结构的动态性会导致 P'和 P 之间存在差异, 最小化这个差异, 也是动态图划分的另一个重要目标. 对于顶点 v 或者边 e, 用 P_v 和 P_e 表示 v 和 e 在划分 P 下所在的分区, 用 P'_v 和 P'_e 表示 v 和 e 在 P'划分下所在的分区, 用集合 $C(P,P')$表示所有 $P_v\neq P'_v$ 和 $P_e\neq P'_e$ 的顶点和边的集合, 动态图划分需要最小化 $\min C(P,P')$.

传统意义上的动态图指的是结构会发生动态变化的图, 如顶点和边的插入和删除, 而图结构的动态变化会破坏图的初始划分. 在真实应用中, 各个实体之间的关系以及实体本身都具有动态性. 例如, 如图 4 所示, 在社交网络中, 我们将每个用户看作一个顶点, 用边表示用户之间的好友关系, 新好友关系的建立、好友关系的终结、新用户的产生、旧用户的注销等, 都会导致图结构的动态变化. 而图结构动态会导致负载不均衡和

通信量的增大.

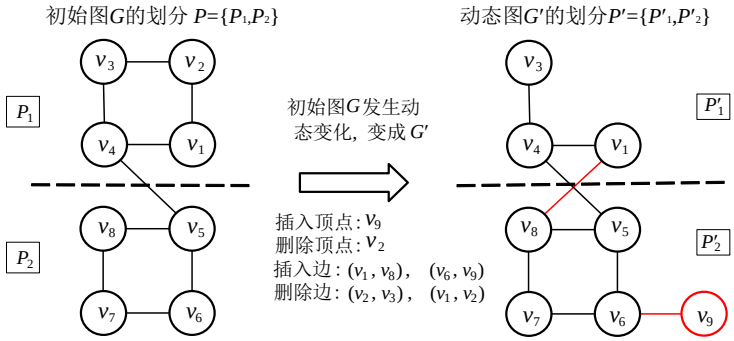


图 4 图结构的动态性

随着分布式图处理系统的发展，分布式图计算领域引申出了另一种动态图，即动态图计算中的图. 图计算任务通常是迭代进行的，而在每次迭代中，各机器中只有部分活跃顶点参与实际的计算. 各机器中活跃顶点数量不一定相同，且随着图计算的进行而动态变化，这导致了图划分结果不匹配图计算的实际开销. 如图 5 所示，这是一次 BFS 的计算过程，深色顶点表示活跃顶点，每个“超步”中各机器的活跃顶点的数量均不同. 因此，图计算动态将导致计算负载的不均衡.

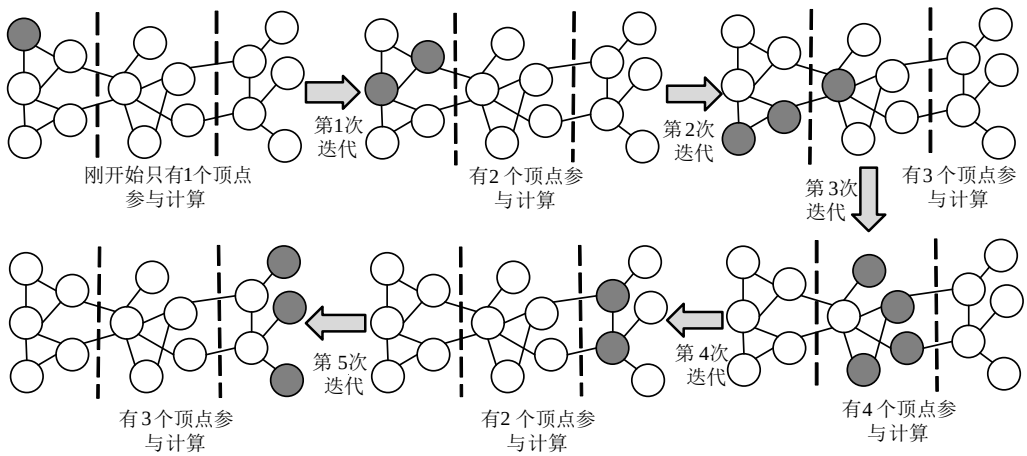


图 5 图计算的动态性

1.4.2 动态图划分问题

对于初始图 $G=(V,E)$，当图发生动态变化变为 $G'=(V',E')$时，将动态变化的那部分图数据标记为ΔG. 动态图划分问题是在原来 G 的初始划分 P 的基础上，处理动态变化图数据ΔG，以获得一个更新后的划分 P'. 当初始划分 P 是顶点划分时，那么 P'需要在维持各个分区中的顶点的数量尽可能相等的同时，最小化边切割 $EC(P)$. 当初始划分 P 是边划分时，那么 P'需要在维持各个分区中的边的数量尽可能相等的同时，最小化点切割 $EC(P)$. 当初始划分 P 是混合割划分时，那么 P'同样需要维持较高的划分质量，即分区负载平衡和最小化不同分区间的通信成本. 同时，对于动态图划分问题来说，除了要保障划分质量外，还需要最小化 $\min C(P,P')$，以减少划分成本和时间开销.

2 流式图划分算法

流式图划分算法被提出的初衷是，以较低的开销划分一个静态图，它将整个图视为一个顶点流或边流，依次将每个顶点或每条边划分到分区中. 流式图划分算法的整个过程如图 6 所示. 流式图划分算法作用于单

个顶点或单条边上, 因此它也可以用于划分动态图中新顶点或新边, 某个时刻的动态图的划分结果可以被视为流式划分中的一个中间状态. 本节我们将流式图划分算法分为基于 Hash 的划分算法、基于邻居分布的划分算法以及基于流的优化划分算法这三大类, 并分别进行介绍, 且在各个子类中分为边割、点割和混合割进行归纳总结.

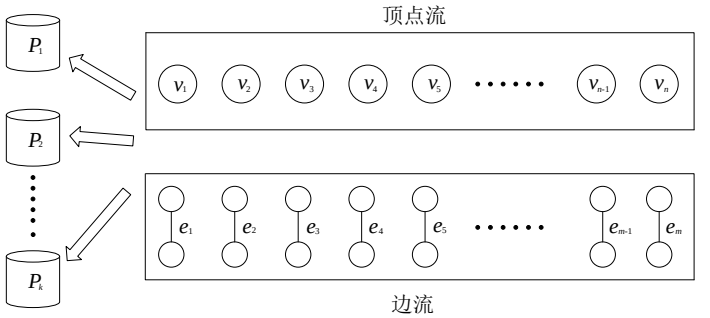


图 6 流式图划分算法思路框架图

2.1 基于Hash的划分算法

基于 Hash 的算法使用一致的哈希函数将顶点或边映射到不同的分区. 例如, 在边划分中, 可以将边与分区间的映射函数定义为下式:

$$f(e)=hash(e) \bmod k \tag{3}$$

在设计合理的哈希函数下, 这可以使分区的负载几乎完全平衡. 基于 Hash 的算法效率较高, 且不需要图结构先验知识, 但它会导致大量的边割或点割.

- 边割

文献[27]提出了一种 Hash 方法, 哈希的好处是可以从集群中的任何机器快速找到每个顶点, 而不需要维护分布式映射表. 它将图中所有顶点根据其 ID 映射到不同的分区中, 其映射函数定义为下式:

$$H(v)=(v \bmod k)+1 \tag{4}$$

- 点割

在 Hash 思想的基础上, 有研究提出了一些改进的策略. DBH[23]是基于 Hash 的边划分算法, 但它进一步考虑了图中顶点的度分布. 真实世界中的图大多是幂律图, 即大多数顶点的度数较低, 而少数顶点的度数较高. DBH 通过优先切割高度的顶点来减少顶点副本. 对于一条边, 它将边分配到度低的端点的 Hash 值对应的分区. 它同样需要维护顶点的度信息, 但是由于 Hash 函数可以即时计算, 所以不需要在分配之前获取图中的全局信息. Grid[24]和 PDS[24]也是基于 Hash 的边划分算法, 它们利用网格来辅助图划分. 它们都先将分区 ID 排列在网格中, 对于每条传入的边, 根据端点的 Hash 值将端点映射到网格中. 再通过获取端点所在的分区的行和列中的所有分区来形成每个末端点的一组受限分区. 然而它们都对分区数量有限制, Grid 需要满足分区数量可以构成一个方阵$|P|=n\times m$ 的条件, PDS 则需要满足(p^2+p+1)个分区, 其中, p 是质数.

- 混合割

Powerlyra[12]提出了平衡 p 路混合切割, 重点是减少低度顶点的复制因子. 具体地, 对于低度顶点, 它采用低切割的方法, 利用 Hash 函数通过对目标顶点进行散列, 将该顶点连同内边分配给机器. 对于高度顶点, 它采用高切割的方法, 利用 Hash 函数通过散列源顶点均匀分配所有的内边给不同的机器. 然后, 混合切割创建副本并构建局部图, 这与典型的顶点切割中所做的操作类似.

2.2 基于邻居分布的划分算法

基于 Hash 的算法完全忽略了图的拓扑结构和历史划分信息, 因此会导致分区间存在大量的边割或点割. 有一些研究利用新到达的顶点和边的邻居信息, 进一步提升了划分质量.

- 边割

LGD[27]是流式顶点划分算法, 它贪心地将顶点划分到邻居最多的分区, 从而最小化局部的边割数, 对应的顶点与分区间的映射函数如下式所示:

$$f(v)=\arg\max_{i\in[k]}\left\{|P^t(i)\cap\Gamma(v)|\left(1-\frac{|P^t(i)|}{C}\right)\right\} \tag{5}$$

其中, $P^t(i)$指的是在 t 时刻分区 i 的顶点集合, $\Gamma(v)$是 v 的邻居集合, C 是分区负载的上限. LGD 通过对高负载的分区进行惩罚来维持分区平衡.

另一个很有代表性的流式顶点划分策略是 Fennel[28], 它的思想与 LGD 非常接近. 其映射函数如下式所示:

$$f(v)=\arg\max_{i\in[k]}\left\{|P^t(i)\cap\Gamma(v)|-\left(\alpha\times\frac{\gamma}{2}\times|P^t(i)|^{\gamma-1}\right)\right\} \tag{6}$$

其中, γ表示分区负载情况在映射函数中占的比重, α则控制分区大小的缩放. 它将图划分问题描述为流环境中的模块度最大化, 并放宽了顶点划分中分区负载的硬约束. 除了经典的 LGD 和 Fennel, 还有一些方法在它们的基础上作了较小的改进. 比如: AKIN[29]用 Jaccard 系数来计算顶点之间的相似性, 而不是仅仅根据邻居数量来分配顶点. 此外, 文献[79]在 LGD 的基础上考虑到集群中网络带宽及节点计算能力的不同, 提出了一种异构感知的流式图划分算法.

- 点割

在流式边划分算法中, Powergraph 提出了一个贪心策略 Greedy[11], 定义了针对不同类型的边的划分规则: (1) 两个端点都是新顶点, 将边分配到负载最小的分区; (2) 一个端点是新顶点, 另一个已经在分区中, 将边分配到该分区中; (3) 两个端点都在分区中, 且存在于同一个分区中, 将边分配到共同分区中; (4) 两个端点都在分区中, 但不在同一个分区中, 将边分配到这些分区中负载最小的分区. 在分布式图处理系统中, Greedy 需要机器之间进行协调, 因此又被分成两个版本.

1) Coordinated: 每个顶点所在的分区拥有全局信息;

2) Oblivious: 每个分区独自管理该分区中顶点, 信息不共享.

Coordinated 的划分质量更高, 但需要更多的时间来进行分区间的通信.

HDRF[32]在 Greedy 的基础上进一步考虑了顶点的度. 对于边 $e(v_i,v_j)$, HDRF 的映射函数如下式所示:

$$f(e)=\arg\max_{p\in P}\{C_{REP}^{HDRF}(v_i,v_j,p)+C_{BAL}^{HDRF}(p)\} \tag{7}$$

C_{REP}^{HDRF} 部分表示的是将边 $e(v_i,v_j)$划分到 p 分区对点切割的影响, 具体如下所示:

$$C_{REP}^{HDRF}(v_i,v_j,p)=g(v_i,p)+g(v_j,p) \tag{8}$$

$$g(v,p)=\begin{cases}1+(1-\theta(v)), & \text{if } p\in A(v)\\ 0, & \text{otherwise}\end{cases} \tag{9}$$

$$\theta(v_i)=\frac{\delta(v_i)}{\delta(v_i)+\delta(v_j)}=1-\theta(v_j) \tag{10}$$

其中, $\delta(v_i)$表示顶点 v_i 的度, $A(v)$表示拥有顶点 v 的副本所在的分区集合. C_{BAL}^{HDRF} 部分表示的是将边 $e(v_i,v_j)$划分到 p 分区对负载平衡的影响, 具体如下所示:

$$C_{BAL}^{HDRF}(p)=\lambda\cdot\frac{maxsize-|p|}{\varepsilon+maxsize-minsize} \tag{11}$$

其中, λ控制分区负载平衡在映射函数中所占的比例, *maxsize* 和 *minsize* 分别是具有最大负载和最小负载的分区的大小. 与 Greedy 相比, HDRF 考虑到幂律度分布, 优先赋值度高的顶点可以进一步减少顶点副本. 此外, HDRF 用参数λ来控制分区负载在整个分数中的权重, 通常设置λ大于 1, 以避免所有边都被分到同一个分区. 而所有边都被分到同一个分区这种情况在 Greedy 中可能会出现.

- 混合割

受到 Fennel 的启发, Powerlyra[12]提出了一种针对低度顶点的流式算法 ginger, 其映射函数如下式所示:

$$\delta g(v,S_i)=|N(v)\cap S_i|-\delta c(|S_i|^V) \tag{12}$$

其中, $N(v)$表示顶点 v 的入边邻居集合, $|S_i|^V$表示分区 S_i 中顶点的数量, 平衡公式$\delta c(x)$表示将顶点 v 加入分区中的成本被用来平衡分区的大小. Ginger 对较低度数的顶点使用类 Fennel 算法对顶点及其入边进行划分. 与 Fennel 算法不同的是, Ginger 的映射函数同时考虑了顶点和边, 顶点的入边跟随顶点一起被分配. 并且, 对于高度顶点, Ginger 使用 Hash 算法对顶点的入边进行划分.

2.3 基于流的优化划分算法

上述基于 Hash 和基于邻居分布的算法均为单元素的流式算法, 即每次划分一个顶点或一条边. 在无法得知后续流信息的情况下, 为了平衡分区的负载, 一些顶点和边会违背贪心策略而被分配到负载较小的分区, 这也是流式算法的划分效果不如静态图划分算法的原因. 为了缓解这个弊端, 一些新颖的流式算法对单元素流的形式进行了改进, 主要可分为 3 种策略: 提前获取图信息、局部优化和分布式流.

- 边割

文献[30]通过提前获取图的局部结构信息来提升划分质量, 提出了基于窗口的流式顶点划分算法, 它在分配每个顶点之前再多收集一些该顶点的信息, 再通过贪心策略将顶点分配到分区中. 文献[80]则通过提前获取图的全局结构信息来提升划分质量, 提出了 re-streaming 的顶点划分思想. 它利用上一次划分结果中的信息来提升这一次图划分结果的质量. 它为 LDG 设计了 re-LGD, 为 Fennel 设计了 re-Fennel, 在映射函数中, 用上一次划分结果中的邻居数量代替原公式中已分配的邻居数量. 此外, Loom[31]将流式图划分运用在图的在线查询上. 有时候, 图的查询是基于模式的, 有些常用的模式的子图被查询的概率更高, 这就使传统的图划分的目标——最小化边割数量失去了作用. 因此, 基于查询模式, 它提供了流式算法将新顶点和新边分配到分区中, 试图将查询频率高的模式分配到同一分区中.

- 点割

Adwise[33]通过提前获取图的局部结构信息来提升划分质量, 提出了基于自适应窗口的流式边划分算法. 它也用窗口来缓存一段时间内到来的边, 从中选择最优的边进行分配. 具体分配策略类似于 HDRF 算法, 但它在此基础上考虑了新边邻居的聚集程度, 更趋向于将局部的边都聚集在同一分区. 2PS[81]则通过提前获取图的全局结构信息来提升划分质量, 是流式边划分算法, 它先通过聚类算法来提前获取图结构信息, 从而指导接下来的流式图划分. RBSEP[82]是边划分算法, 它同时采用了缓存和重分配策略. 在 HDRF 算法的基础上, 它将由于分区负载而违背了贪心策略的新边放入缓存, 延迟分配. 若延迟分配依然无法满足贪心策略, 则利用重分配策略从超载分区中移出部分边, 使缓存中的边能够被划分到最优的分区. 文献[83]则提出了分布式边划分算法, 它将边划分问题建模为一个博弈过程, 边的分区选择被视为游戏中玩家的合理策略选择, 并证明博弈过程中存在纳什均衡. 在具体算法中, 整个边流被分成一系列批次, 各分区并行地对一批边进行选择. 此外, HoVerCut[34]认为, 流式边划分算法在大规模的分布式集群中会降低效率, 因此它提出了并行的分布式流式图划分, 利用多个线程对图进行划分. 每个线程都运行相同的流式图划分算法, 并维持一个该分区的状态信息. 状态信息只是该分区内的顶点和边的信息, 是部分的且不完整的. 然后, 通过一个共享的全局状态信息, 使每个线程可以获得全局状态. 为了降低大规模分布式带来的延迟, 它将共享状态信息分享给一个窗口中的边而不是每一条边.

- 混合割

为了在分布式图存储中实现低延迟和高吞吐量的在线查询, 文献[84]提出了一个新的工作负载自适应和拓扑感知的流划分方法 WASP. 由于每个查询工作负载通常包含流行的或类似的查询, WASP 捕获在现有查询工作负载中经常访问和遍历的活动顶点和边. 使用这些信息, 通过避免活动顶点按访问频率的比例集中在几个分区中, 或者通过降低切割的概率, 可以提高划分的质量. 具体地, 对于工作负载自适应, WASP 逐渐调整与现有查询工作负载中频繁遍历的活动边和边中频繁探索的端点活动顶点有关的分区. 对于拓扑感知, WASP

利用混合切割模型, 将高度顶点的邻边均匀分配都不同的分区. 对于每个顶点, 当顶点的度(入边和出边的度)超过一个可配置的分割阈值时, 它被认为是一个高度顶点, 从而导致该顶点的边分裂.

表 1 给出了流式动态图划分算法的一个小结.

表 1 流式图划分算法小结

类型	优点	缺点	划分方式	算法	特点
基于哈希的划分算法	算法效率高, 不需要图结构先验知识和历史划分信息	会导致大量的边割或点割	边割	文献[27]	根据每个顶点的 ID, 将其映射到不同的分区中
			点割	DBH[23]	将边分配到度低的端点的 Hash 值对应的分区, 优先切割高度顶点
				Grid, PDS[24]	将分区 ID 排列在网格中, 每条边根据端点的 Hash 值, 将端点映射到网格中. 利用网格辅助图划分
			混合割	Powerlyra[12]	利用 Hash 对低度顶点进行散列, 将该顶点连同内边分配, 利用 Hash 散列高度顶点均匀分配所有内边给不同分区
基于邻居分布的划分算法	利用图的拓扑结构和历史划分信息, 相比基于 Hash 的方法获得更好的划分质量	无法得知后续流信息, 一些顶点或边会违背贪心策略被分配到负载较小的分区, 从而造成冗余的边割或点割	边割	LGD[27]	贪心地将顶点划分到邻居最多的分区中, 以最小化局部的边割数
				Fennel[28]	它的思想与 LGD 非常接近, 但放宽了分区负载的硬约束
				AKIN[29]	不仅根据邻居数量, 还利用 Jaccard 相似度作为辅助信息分配顶点, 效率不高
				文献[79]	在 LGD 的基础上考虑集群中网络带宽及节点计算能力的不同
			点割	Greedy[11]	使用贪心策略, 针对不同类型的边设计不同的划分规则
				HDRF[32]	在 Greedy 的基础上考虑顶点的度, 优先处理高度顶点
			混合割	Ginger[12]	利用类 Fennel 对低度顶点及其入边划分. 利用 Hash 对高度顶点的入边划分
基于流的优化划分算法	通过提前获取图信息、局部优化或者分布式流的策略, 相比其他方法进一步提高了划分质量	由于辅助手段会造成额外的时间或者空间开销, 算法效率相比其他方法要低	边割	文献[30]	通过基于窗口的策略获取将要分配顶点的局部信息, 以提高划分质量
				文献[80]	利用上一次划分结果作为全局信息参与这一次的划分
				Loom[61]	适用于在线查询图的划分需求, 将查询频率高的模式分配到同一分区
			点割	Adwise[33]	具体分配策略类似于 HDRF, 但它更趋向于将局部的边都聚集在同一分区
				2PS[81]	通过聚类算法提前获取全局信息指导后面的划分
				RBSEP[82]	在 HDRF 的基础上, 同时采用了缓存和重分配策略
				HoVerCut[34]	利用多个线程并行执行多个流式划分算法, 效率较高
				文献[83]	提出并行的分布式算法, 将问题建模为一个博弈过程
			混合割	WASP[84]	对于工作负载自适应, 利用原有的查询信息不断调整分区. 对于拓扑感知, 采用混合切割策略

流式图划分算法的特点是将图看成一个顶点或者边序列, 不需要将整个图放到内存中进行划分, 因此它用于划分大规模图数据具有较高的效率. Hash 算法是一种简单而有效的方法, 尤其是在降低延迟方面. 在基于邻居分布的算法中, 几乎所有的算法在设计映射函数时除了考虑邻居信息之外, 也将分区负载平衡作为一项考虑其中. LGD 和 Fennel 算法都是顶点划分, 更适用于对不遵循幂律分布的图进行划分. LGD 与 Fennel 类似, 但是 Fennel 相比较 LGD 放宽了分区负载的硬约束. AKIN 算法在 LGD 和 Fennel 的基础上通过考虑顶点之间的相似性, 而不仅仅是单个顶点的邻居分布, 以进一步提高图划分的质量. 但是 AKIN 算法计算相似性时并没有考虑顶点的度分布, 并且需要一定计算开销. 文献[79]更适合用于异构环境下的图划分. Greedy 和 HDRF 都是边划分算法, HDRF 在 Greedy 的基础上进一步考虑了顶点的度分布, HDRF 算法更适合用于划分具有幂律分布的大规模动态图. 此外, Ginger 算法针对具有幂律分布的真实图, 结合了边切割和点切割这两种划分方式的优势, 即边割具有更好的局部性和点割具有更好的负载平衡, 对高度顶点和低度顶点分开考虑, 提出了混合切割策略, 因此, Ginger 算法适合划分幂律图. 一些新颖的流式算法采用提前获取图信息、局部优化和分布式流这 3 种不同的策略进一步改进图划分的质量. 基于邻居分布的算法相比较 Hash 算法具有更好的划分质量, 但是却比 Hash 算法效率低一些. 而新颖的流式划分算法由于要获取辅助信息以及进行重分配等需要额外的时间和空间开销, 因此在提升划分质量的基础上, 同时也损失了一定的划分效率. 文献[30]和 Adwise 都是利用窗口获取局部信息进行划分, 而文献[80]和 2PS 都是利用全局信息指导划分. HoVerCut 和文献[83]是并行

的分布式算法, 具有较高的划分效率. RBSEP 和 WASP 都用到了重分配的思想, 以不断改进图划分的质量, 但 WASP 更适合划分幂律分布的图. Loom 算法适合用在图数据库的在线查询上.

3 增量式图划分算法

动态图的总体规模通常是扩大的, 其中伴随着顶点和边的插入与删除, 且动态图是无界的. 流式图划分算法虽然可以被用于动态图划分, 但具有一定的局限性: 一方面, 流式图划分算法只能处理新顶点或新边, 无法处理图中发生的删除; 另一方面, 流式图划分算法通常不支持对已划分部分的调整, 随着图规模的扩大, 其划分质量较低的弊端将被放大. 针对这些特点, 有一些研究提出了增量式动态图划分算法, 采用不同的策略对不同类型的动态变化进行划分. 这类算法适合于持续增长的动态图. 本节将从单元素增量式划分和批处理增量式划分这两大类分别进行介绍, 并且同样在各个子类中从边割、点割以及混合割不同的划分方式进行归纳总结.

3.1 单元素增量式划分

有一些研究为每个单独的动态变化提供了划分策略.

- 边割

文献[85]提出了基于边割的增量式图划分算法, 为插入顶点、插入边、删除顶点和删除边分别提供了增量式的策略. 在插入顶点时, 除了考虑邻居分布和分区负载之外, 还进一步将分区间的单位通信代价和原分区间边割数加入到映射函数, 从而感知非均匀的网络通信代价. 在删除顶点时, 若出现分区欠载的情况, 则从负载高的分区转移边到欠载分区. 在插入新边时, 若端点的邻居分布导致了冗余边割, 则将端点转移至邻居最多的分区. 在删除边时, 将外部邻居数大于内部邻居数的端点转移至对应的分区.

- 点割

GR-DEP[86]则是单元素的增量式的边划分算法, 它在插入新边或删除边后对局部结构进行调整, 从而减少局部的顶点切割. 具体地, GR-DEP 提出了组团边, 将多条相连的边视为一个整体, 通过计算组团边与各分区之间共享顶点数的差异, 可以判断组团边是否需要被重分配, 并提出了组团边搜索算法, 利用结构感知的优先级策略探索动态边附近的结构. 由于插入的新边有可能被包含在组团边中而被重分配, 因此, GR-DEP 在插入新边时预测该边在重分配后所在的分区, 从而减少不必要的开销. GR-DEP 还提出了局部次优划分的概念, 统一地定义了边划分中存在的冗余顶点副本, 并证明了随着图规模的扩大, 在不调整分区结构的情况下, 动态图的划分质量会逐渐下降. 此外, GR-DEP 还提出了完整的分布式动态图划分框架(如图 7 所示), 动态变化以分布式流的方式进入分区中, 每一个分区都有一个动态划分器和一个 EG 重划分器. 动态划分器用于处理每一个到达的动态变化, EG 重划分器在动态变化附近搜索组团边对其进行重分配. 每一个顶点持有一个邻接列表和拥有该顶点副本的分区集合, 并且每条边由两个端点连接.

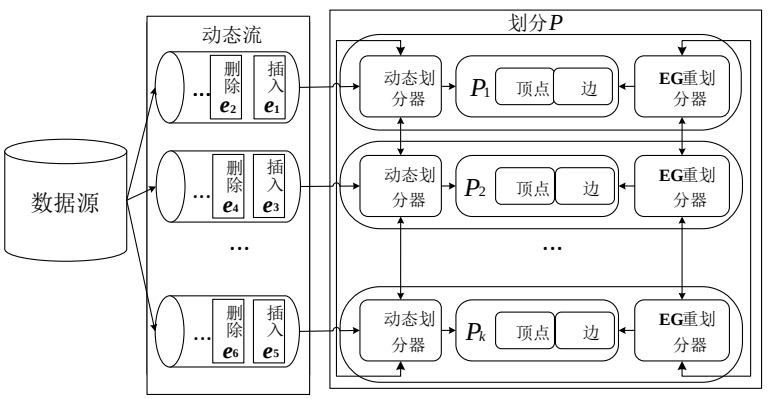


图 7 GR-DEP 的分布式动态图划分框架

DynamicDFEP[87]在 DFEP 的基础上提出了动态图划分的策略. DFEP 是基于资金的静态边划分算法, 每个分区随机选择一个顶点, 并为其分配初始“资金”. 每个顶点使用资金尝试“购买”邻边, 购买成功后, 资金将被传递到邻居顶点, 分区在购买的过程中逐渐扩展. 当分区的资金不足时, 根据不同的负载, 分区会被给予额外的资金. DynamicDFEP 规定, 动态图中的每条新边都被分配到在那条边上投入资金最多的分区, 而当顶点或边被删除且导致分区负载不均衡时, 用 DFEP 重新划分整个图. DynamicDFEP 的全局架构如图 8 所示, 用户程序的副本在一个机器集群上执行, 其中一个作为 master 决定分区数量并且执行更新函数, 负责协调多个 worker. 具体地, 初始输入图是由 DFEP 算法划分, 当 master 执行图结构的一个动态更新时, 增量式计算开始. 为了更新图数据, master 根据 Hash 函数将动态更新数据分配给 workers. 每一个 worker 收到数据后开始更新. 当 worker 终止时, 将发送一个消息给 master, 然后, master 衡量分区是否平衡, 或者是否需要重划分.

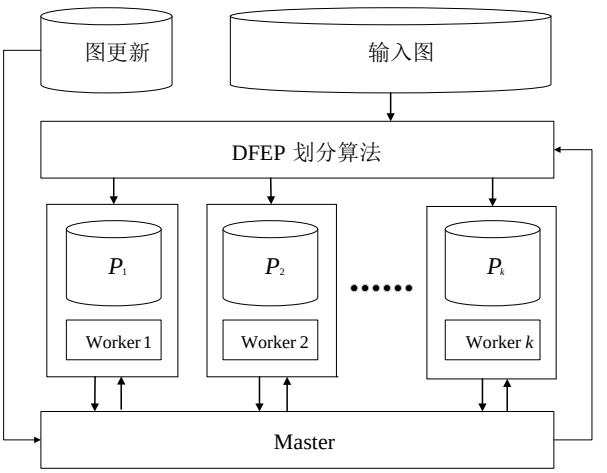


图 8 DynamicDFEP 的框架图

- 混合割

IOGP[88]提出了一种用于分布式图数据库的增量式图划分算法, 它结合了边切割和点切割, 其中, 点切割只用来处理高度顶点, 而高度顶点是由用户定义的阈值控制. 具体来说, IOGP 首先利用确定性 Hash 算法快速分配新顶点, 默认情况下, 该策略允许对大多数顶点进行单跳访问. 当一个顶点插入更多的边时, IOGP 再利用 Fennel 算法的思想, 通过顶点的转移来优化分区, 调整顶点的位置, 以利用不断增长的关于顶点连接性的知识实现更好的局部性. 在此步骤之前, 图仍然利用边切割进行划分. 但是, 一旦一个顶点有太多的边, IOGP 将利用点切割来增加并行度, 进一步提高划分和遍历性能. Leopard[89]提出了具有复制可能性的动态边割划分, 即结合边割和点割的混合动态划分算法. Leopard 也是单元素增量式图划分算法, 它在插入新顶点时对局部结构进行调整, 从而减少边割数量. 具体地, 它利用 Fennel 将顶点分配到分区中, 然后在该顶点的邻居中寻找没有处于最优分区的顶点, 并将它们转移到邻居最多的分区中. 此外, 它还通过创建顶点副本使顶点可以本地访问邻居, 并根据图任务是否只读以及主顶点和副本顶点的位置信息, 最小化副本数量.

3.2 批处理增量式划分

另一些研究则提出了针对一批动态变化的划分策略, 与单元素增量式划分算法相比, 收集一段时间内的动态变化对划分质量的提升更有利, 但牺牲了实时性.

- 边割

文献[90]提出了基于线性规划的增量式图划分方法. 它首先将新顶点都分配给最近的顶点所在的分区, 这会导致分区负载不均衡, 需要进一步通过顶点转移来维持负载均衡, 并同时最小化通信代价. 接着, 它将边界顶点的邻居所在的分区作为其潜在的转移目标分区, 并以此作为各分区间的转移量上限. 然后, 它将最小

化分区改变程度表述为一个线性规划问题:

$$\left.\begin{aligned} &Minimize \sum_{0\leqslant i\neq j<k} l_{ij} \\ &\text{s.t. } 0\leqslant l_{ij}\leqslant a_{ij}\leqslant |P_i| \\ &\sum_{0\leqslant i<k}(l_{ij}-l_{ji})=|P_j|-\mu,\ 0\leqslant j<k \end{aligned}\right\} \tag{13}$$

其中, l_{ij} 是从 P_i 转移到 P_j 的顶点数, a_{ij} 则是 P_i 中可以被转移到 P_j 的顶点数, μ是平均分区负载. 利用单纯形法求解线性规划问题, 得到各分区间的转移量并完成转移操作. 最后, 通过另一个线性规划优化分区间的通信代价:

$$\left.\begin{aligned} &Minimize \sum_{0\leqslant i\neq j<k} l_{ij} \\ &\text{s.t. } 0\leqslant l_{ij}\leqslant b_{ij},\ 0\leqslant i\neq j<k \\ &\sum_{0\leqslant i<j}(l_{ij}-l_{ji})=0,\ 0\leqslant j<k \end{aligned}\right\} \tag{14}$$

其中, b_{ij} 是 P_i 中在 P_j 中邻居更多的顶点数量. 最终, 迭代地求解此步骤, 直到通过顶点移动获得的有效增益很小为止. 然而, 单纯形法的时间复杂度是指数级的, 难以运用于大规模图上.

- 边割和点割

最近, 文献[26]定义了批处理增量式图划分问题: 一个迭代式图划分算法 A(如 NE、DNE、KGGGP、FENNEL、HDRF 等)可以被视为一个更新函数 f 和一个范围函数 h 的结合, f 用于在每次迭代中划分图结构并更新分区状态, h 则用于获取下一次迭代即将被划分的部分. 用 D_A^t 表示算法 A 在第 t 轮迭代中生成的分区的状态, 则 D_A^{t+1} 可以通过 f 和 h 来得到:

$$\left.\begin{aligned} &D_A^{t+1}=D_A^t\oplus f(H_t) \\ &H_t=h(D_A^t) \end{aligned}\right\} \tag{15}$$

用ΔA 表示算法 A 的增量式版本, 用于划分图 G 上的一批动态变化ΔG, 则根据 f 和 h, 可以在不改变初始分区的情况下划分ΔG. 文献[26]中提出了高质量静态图划分算法 DNE 和 KGGGP 的增量式版本 IncDNE 和 IncKGGGP 来处理动态变化.

DNE 是目前效果最好的边划分算法 NE 的分布式版本, NE 通过邻居扩展的策略顺序地生成每个分区, DNE 则并行地生成每个分区. 在分区生成过程中, 根据邻居策略选择在分区外邻边最少的顶点 x, 邻居策略公式如下式所示:

$$x:=\arg\min_{v\in S\setminus C}|N(v)\setminus S| \tag{16}$$

其中, S 是分区中的顶点集合; C 是核心顶点集合, 即所有邻边都已在分区内的顶点集合. x 的所有邻边将被分配到分区, 并将 x 加入该分区的核心顶点集合, 直到分区满载. DNE 的更新函数 f 即邻居扩展策略, 范围函数 f 则标识每个新选择的边界顶点 v 的未分配的邻边, 并从 v 所在的分区中导出它们的候选分区. 为了维持分区负载均衡, IncDNE 从超载分区中移除部分边加入到ΔG 中; 同时, 为了进一步提升划分质量, 若ΔG 中的顶点 v 在分区中只有少量邻边, 那么 IncDNE 将 v 的邻边从各分区中移除并加入到ΔG 中, 因为它们被视为先前的划分缺少关于 v 的完整信息, 重分配有利于提升划分质量.

最终, IncDNE 在初始分区的基础上, 用 f 和 h 来划分ΔG.

KGGGP 是分布式的顶点划分算法, 它利用边界扩展策略并行地生成每个分区. 它从随机选择的种子顶点 vr 开始, 并不断地从边界顶点的邻居中通过分区映射函数来迭代地扩展每个分区 P_i, 其映射函数如下式:

$$f(v)=\arg\max_{P_i\in P}\sum_{v'\in P_i,(v,v')\in E}|v'|-v.unalloc \tag{17}$$

KGGGP 的更新函数 f 即边界扩展策略, 范围函数 f 则调整其邻居的得分值, 并发起对未分配邻居的分配请求. 与 IncDNE 类似, IncKGGGP 也从分区中移除一部分顶点加入ΔG 中, 然后用 f 和 h 来划分ΔG.

表 2 给出了增量式动态图划分算法的一个小结. 文献[85]是针对顶点划分提出来的单元素增量式图划分算法, 重点针对4种动态更新提出不同的划分策略, 即顶点的插入和删除以及边的插入和删除. 在单元素增量式边划分中, GR-DEP 主要用于实时环境中的具有幂律分布的真实动态图划分. 而 DynamicDFEP 算法当出现边或者顶点的删除以及出现负载不平衡时, 会重新用 DFEP 算法划分整个图, 因此具有较低的效率. Leopard 和 IOGP 结合了边割和点割的优势, 采用了混合切割的策略, 更适合于具有幂律分布特点的真实图. Leopard 和 IOGP 都用到了重分配的思想, 并且是轻量级实现, 因此拥有更高的划分质量和划分效率. IOGP 主要用于分布式图数据库的在线查询. 文献[26,90]都是批量式增量图划分算法, 它们都是收集一段时间内的动态更新, 然后集中式地对一段时间内的动态更新进行处理, 因此这些算法缺乏实时性. 文献[90]主要通过将问题转换为线性规划问题, 然后通过传统的最优化方法求解. 由于文献[90]需要进行数学计算, 存在着性能的瓶颈, 因此相比较而言, 不能够用来划分大规模的动态图数据. 文献[26]主要是从已有的具有较高划分质量的静态图划分算法中发展而来, 对划分质量的界限有一定的保障. 由于文献[26]所提出算法的特点, 具有较低的开销, 它可以被用来处理大规模动态图数据.

表 2 增量式图划分算法小结

类型	优点	缺点	划分方式	算法	特点
单元素增量式划分算法	具有实时性, 满足时间敏感的图应用的划分需求	处理单个动态变化时无法获知其局部动态变化, 可能会影响划分质量	边割	文献[85]	为插入顶点和边, 删除顶点和边分别提出处理策略, 通过移动单个顶点作局部调整
			点割	DynamicDFEP[87]	当顶点或边被删除导致分区负载不平衡时, 用 DFEP 重划分整个图, 效率不高
				GR-DEP[86]	为插入边和删除边提供处理策略, 通过移动组团边作局部调整
			混合割	Leopard[89]	利用 Fennel 将顶点分配到分区中, 并对局部结构进行调整. 适合于只读图计算任务
				IOGP[88]	先利用 Hash 分配顶点, 再利用 Fennel 的思想通过转移顶点优化分区. 对于高度顶点采用点割划分策略. 适用于图的在线查询
批处理增量式划分算法	由于获知局部动态变化, 利用其信息将进一步提高划分质量	缺乏实时性	边割	文献[90]	将问题表述为线性规划问题利用最优化方法求解, 存在性能瓶颈
			边割、点割	文献[26]	将已有具有迭代特点的静态图划分算法增量化, 对划分质量有一定保障

4 图重划分算法

在第 1.2 节中, 本文介绍了图的动态性主要有两个来源: 图结构的动态性和图计算的动态性. 流式图划分算法和增量式图划分算法主要针对图结构的动态性, 即 $G'=G+\Delta G$, 用于划分图中的动态部分ΔG. 重划分算法则专注于在图动态变化后对分区进行优化, 通过分区间顶点或边的转移来减少分区间的通信代价, 以及维持分区负载的平衡. 因此, 重划分算法既可以处理结构动态图, 也可以处理计算动态图. 本节分为针对结构动态的重划分算法和针对计算动态的重划分算法进行归纳总结, 并在各个子类中用不同的划分方式对算法进行归类介绍.

4.1 针对结构动态的划分算法

当图发生动态变化后, 即顶点或边的插入或删除, 初始图的划分质量会随之下降, 进而影响图计算和图处理的性能, 因此, 一些研究者提出了重划分算法以解决这类问题.

- 边割

在基于边割的图划分环境中, 顶点通常需要被分配到邻居最多的分区. 如图 9 所示, 因为顶点 a 在 P_2 的邻居多于在 P_1 的邻居, 因此将顶点 a 转移到 P_2 可以减少边割, 可以定义一个顶点 v 从 P_i 转移到 P_j 的增益为

$$gain(v)=d_v(j)-d_v(i) \tag{18}$$

其中, $d_v(i)$表示顶点 v 在分区 P_i 中的邻居数量. 重划分就是将 $gain$ 大于 0 的顶点转移到目标分区.

KL[55]首先提出了重划分思想, 它不断地在两个分区间寻找顶点对(v_1,v_2), 使它们交换分区的增益大于 0,

这需要消耗 O(n^2log$n$)的时间. FM 算法[56]则对 KL 算法进行了改进, 它是在分区中寻找增益大于 0 的单个顶点, 只需要线性的时间开销. 这两种算法允许顶点交换或转移的增益为负, 这可能使之后的总增益为正, 从而逃避局部最优. 重划分的思想被广泛地用于图划分中, 例如 Metis、Parmetis 等多级图划分算法在反粗糙化阶段使用 KL 和 FM 算法来减少边割. jabeja[53]也使用了重分配的思想, 它在完全分布式的环境中, 首先随机为每个顶点分配分区, 然后通过分区之间顶点的交换来减少边割. 这些算法虽然用到了重划分思想, 但它们的本质依然是静态图划分算法, 重划分只是其中的一个步骤.

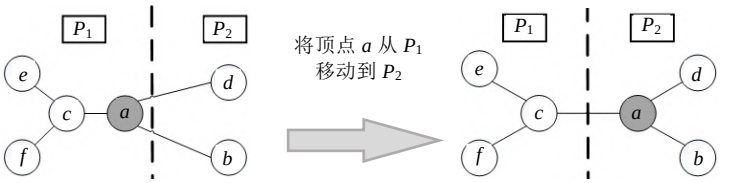


图 9 基于边割的重划分

有一些研究则针对动态图提出了重划分算法. 文献[91]提出了多级图的重划分算法, 在初始划分 P 的基础上, 通过粗化、多级扩散和多级优化这 3 个阶段来获取新的划分 P'. 在粗化阶段, 各分区通过将多个顶点聚集成一个大顶点的方式来逐级地缩小分区规模, 最终达到一个粗化的图. 在扩散阶段, 粗化地图将逐渐细化, 即大顶点被拆成多个小顶点, 并通过分区间的顶点转移来维持分区负载均衡. 在优化阶段, 各分区继续逐级细化, 并通过顶点的转移来减少边割数量.

由于多级图划分算法不适合于大规模图, 近期有许多轻量级的重划分算法被提出. 它们只需要少量关于图的信息, 在各个分区中迭代地寻找 *gain* 大于 0 的顶点进行转移, 直到无法优化. xdgp[92]在每次迭代中利用标签传播将顶点转移到邻居多的分区, 等同于将 *gain* 大于 0 的顶点进行转移. 为了平衡分区的负载, 它为每两个分区计算它们之间可以进行转移的最大数量. 但是每个顶点都单独做转移决定, 因此可能会出现转移的震荡. 为了保证算法可以收敛, 它为每个顶点设置了转移概率 s, 当 s=0 时, 顶点无法进行转移. Hermes[35]是应用在图数据库 Neo4j[93]上的重划分算法. 在每一次迭代中, 每个分区并行地寻找 *gain* 大于 0 的顶点, 并从中选出 top-k 个 *gain* 最大的顶点转移到目标分区. 在转移过程中, 若原分区欠载或目标分区超载, 则无法进行转移. 为了保证算法收敛, 它将一次迭代分成两个阶段: 第 1 阶段只允许顶点从 ID 小的分区转移到 ID 大的分区, 第 2 阶段相反. 但是因为 Hermes 会同时并行移动多个顶点, 会存在移动干扰, 从而导致性能下降. 因此, 文献[94]提出了一种有效的两阶段方法以解决这个问题: 第 1 阶段, 在划分边界区域搜索组团点; 第 2 阶段, 找出拥有最大移动增益的组团点进行移动, 不断改进图划分的质量. 具体地, 文献[91]提出了组团点的概念, 通过将紧密连接的一组顶点视为一个整体(如图 10 所示), 其中任意一个顶点单独的移动增益均小于 0, 而整体的移动增益则大于 0. 此外, 文献[94]不仅考虑了顶点转移的边割增益, 还同时考虑了负载增益, 通过权重控制负载和边割对于整体优化目标的影响.

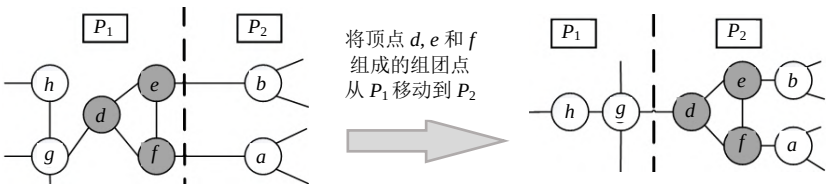


图 10 基于组的边割重划分

上述方法都假设图划分的环境是同构的且无网络争用, 也就是说各分区之间的单位通信代价都一样. Aragon[95]、Paragon[96]和 Planar[97]提出了在异构环境中的图划分, 它们认为: 分区间的单位通信代价是不同的, 在这样的环境下, 仅仅通过邻居数量来计算转移的增益是不准确的, 应该加入分区间通信的权重信息. 它们

从 3 个方面考虑了顶点 v 从 P_i 转移到 P_j 的增益: (1) 顶点 v 的转移对 P_i 和 P_j 之间通信代价的影响; (2) 顶点 v 的转移对其他分区中 v 的邻居与 v 之间通信代价的影响; (3) 转移顶点 v 造成的转移代价. 它们的不同之处在于: Aragon 每次选择两个分区进行顶点的转移; 而 Paragon 选择了一个主分区, 通过主分区将剩下的分区分组, 各组并行地进行 Aragon, 然后各组之间交换分区, 再进行 Aragon; Planar 的策略与 xdgp 类似, 每个分区各自寻找 *gain* 大于 0 的顶点, 然后根据各分区的负载为每个分区分配配额, 即每个分区能够转出或转入的最大顶点数量. 在配额的范围之内, 各分区之间转移顶点.

- 点割

在基于点割的图划分环境中, 重划分通过转移边来减少顶点的副本. 如图 11 所示, 通过将边(a,c)和(c,b)转移到分区 P_2, 可以减少一个顶点副本. 在文献[98]中, 首次提出了针对动态图的重划分. 文中定义了组团边 *GE*, 如图 10 所示, 图中左图的 $GE=\{(a,c),(b,c)\}$, 通过转移 *GE* 可以减少顶点副本. 在实时动态的场景中, 它在分配完新边后立刻检查新边附近是否存在 *GE*: 若有, 则将它转移到目标分区来减少副本. 同时, 为了减少重划分中的转移代价, 文中还提出了一种将新边分配到分区中的策略. 与流式图划分不同, 这种策略的目的是减少重划分中的转移代价, 而不是最小化顶点副本. 这种策略通过预测新边在重划分后所在的分区, 直接将新边分配到该分区, 从而减少重划分的转移代价.

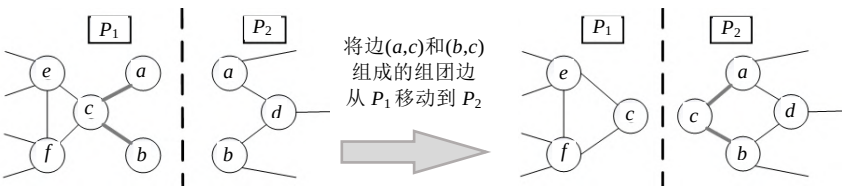


图 11 基于点割的重划分

除此之外, 还有一些基于混合切割的流式图划分方法也使用了重划分思想, 例如: Leopard 和 IOGP 在分配完顶点之后, 对局部结构进行调整, 使分区间的通信代价进一步减少.

4.2 针对计算动态的划分算法

在分布式图处理系统中, 图计算大多都是迭代进行的, 在每个超步中, 每个分区都有一部分顶点被激活而进行计算. 但是随着图的遍历, 被激活的顶点的规模会发生动态变化, 例如单源最短路径计算、广度优先搜索、最大匹配算法等. 不同规模的激活顶点会导致分区中计算负载的不均衡, 同时还会导致分区间通信代价与边割或点割数量不一致, 从而导致原始划分不契合图计算模型. 为了提升图计算的效率, 需要在图计算中进行动态重划分.

- 边割

GPS[99]提出了关于重划分的 3 个关键问题: (1) 哪些顶点要重新分配; (2) 如何以及何时将重新分配的顶点移动到新的分区; (3) 如何找到重新分配的顶点. 顶点在图计算中会向邻居顶点发送消息, 如果顶点 u 与分区 P_i 进行交互的消息比从其他任何分区都多, 则顶点 u 理论上应该被重新分配到 P_i, 以最小化分区间的通信量. 然而, 这会导致分区的负载不均衡, 因此, GPS 在分区之间交换顶点. 每个分区都维护一个集合包含潜在的待重分配的顶点, 并在保证分区中顶点数量不变的前提下交换集合中的顶点. 此外, 为了降低重分配导致的网络开销, GPS 规定: 待交换的顶点在每个超步结束时先标记邻接列表, 在下一个超步参与计算之后再物理转移, 且仅发送邻接列表和顶点最新值, 超步中的消息不会被发送. 当顶点 u 被分配到新分区后, 其他分区需要记录该信息, 这会导致额外的开销. GPS 通过交换顶点 ID 的方式来减少存储开销.

图计算的动态性导致的计算性能下降体现在很多方面, 通常, 重划分算法需要首先检测性能下降的来源, 再针对性地通过顶点或边的转移来提升性能. 在基于边割的图划分环境中, Mizan[100]为了平衡分区负载, 监视每个顶点的 3 个主要信息: 输出消息数、输入消息数、处理消息时间. 这 3 个信息中任意一个值较高, 都可能表示分区不平衡, 需要通过转移部分顶点来使分区平衡. 它首先将统计信息与正态分布进行比较, 识别不

平衡的来源; 然后确定顶点转移的目标是平衡输出消息, 还是平衡输入消息, 或者是平衡处理消息时间; 接下来, 将负载高的分区与负载低的分区一一配对, 从负载高的分区中选择顶点转移到负载低的分区. 然而, Mizan 只考虑了如何平衡负载, 忽略了分区间的通信代价.

CatchW[101]在平衡分区负载的同时, 尝试着去减少分区间的边割. 它利用尺寸变化的窗口来研究图计算中各分区的工作负载. 它也先计算分区之间具体转移的量, 然后在分区中寻找可以减少通信代价的顶点. LogGP[37]认为: 有一些重划分算法在新的图计算环境中需要再次对图进行重划分, 如 CatchW, 但分区结果却没有改进. 因此, 它利用运行统计信息或历史分区日志来进行重划分. 首先, 它将上次图划分的结果和上个超步中活动顶点集合的信息加入到图中, 形成超图; 然后, 对超图进行流式划分, 先分配引脚, 再分配顶点; 最后, 预测下一个超步中的活动顶点, 并评估下一个超步的通信代价. 将这些顶点移动到合适的分区来降低未来的总的通信成本, 并平衡每个节点的运行时间. 这种利用历史信息的想法首先可以使原先在同一分区内的顶点更趋向于在同一分区, 其次可以将一些连接程度较高的顶点分配到同一分区.

- 点割

在基于点割的图划分环境中, GrapH[36]中提出了重划分算法 H-adapt 来提升分区质量. 在异构环境中, 它通过转移一组边来减少分区间的通信量. 依次检查每个边界点的所有邻边是否可以通过转移减少通信代价: 如果是, 则加入到 bag 中. H-adapt 是用于解决图计算导致的负载不平衡以及分区间增加的通信量, 因此无法解决真实世界中的动态图问题. 除此之外, 在方法上, H-adapt 在构建 bag 的过程中只考虑了边界点的邻边, 且默认同时考虑所有邻边, 而忽略了二跳的邻居以及部分邻边的情况, 实际上, 它无法准确地发现能够减少通信代价的边. 除了 H-adapt, GraphSteal[102]也使用重划分算法, 不过它只是用于平衡各分区在图计算过程中出现的运行时间的不平衡. 它将边从较慢的分区转移到较快的分区, 从而提升图计算效率, 但其也忽略了分区间的通信代价.

表 3 给出了动态图重划分算法的一个小结.

表 3 图重划分算法小结

类型	优点	缺点	划分方式	算法	特点
针对结构动态的划分算法	处理图结构的动态变化, 提高图划分的质量	由于顶点或边的迁移, 会造成额外的时间和空间开销	边割	文献[91]	多级图的重划分算法, 适合较小规模的图划分
				xdgp[92]	利用标签传播, 将顶点转移到邻居多的分区作局部调整
				Hermes[35]	通过移动多个增益最大的顶点优化分区, 更适合于分布式图数据库
				文献[94]	解决 Hermes 的移动干扰问题, 通过移动组团点优化分区, 具有较高的划分质量
				Aragon[95]	考虑异构环境, 每次选择两个分区进行顶点的转移优化分区
				Paragon[96]	考虑异构环境, 选择一个主分区, 通过主分区将剩下的分区分组, 各组并行运行 Aragon
				Planar[97]	考虑异构环境, 每个分区各自寻找增益大于 0 的顶点, 在配额的范围内转移顶点优化分区
			点割	文献[98]	通过移动组团边优化分区进一步改进划分质量, 算法效率较高
针对计算动态的划分算法	能够使得特定图计算应用迭代过程中负载更平衡, 通信代价更低	顶点或者边的重分配会造成额外的开销	边割	GPS[99]	每个分区都维护包含潜在待重分配顶点的一个集合, 在分区负载不变的前提下交换集合中的顶点优化分区
				Mizan[100]	考虑平衡分区负载, 首先识别不平衡来源, 然后确定顶点转移的目标, 最后从高负载分区转移顶点到低负载分区
				CatchW[101]	在平衡分区负载的同时, 寻找可以减少通信代价的顶点, 尝试减少分区间的边割
				LogGP[37]	利用运行统计信息或历史分区日志来进行重划分优化分区
			点割	GrapH[36]	通过转移一组边, 减少分区间的通信量, 以提高划分质量
				GraphSteal[102]	适合于平衡各分区出现的运行时间不平衡, 将边从较慢的分区转移到较快的分区

动态图重划分算法分为针对图结构动态的重划分算法和针对图计算动态的重划分算法.

- 在针对图结构动态的重划分算法中, 文献[91]、xdgp、Hermes 和文献[94]都是顶点划分算法. 文献[91]是在多级图划分思想的基础上提出了动态图划分算法, 故该算法只能处理较小规模的动态图数据,

不能处理超大规模的动态图. Hermes 主要是用于支持图数据库的操作, 但是它可能存在顶点转移震荡, 因而可能会导致较差的划分质量和较低的划分效率. 文献[94]主要针对Hermes存在的弊端, 提出了一种新的序列化的方法, 并且不再转移单个顶点, 而是转移由一系列紧密联系的顶点组成的组团点, 移动增益既考虑了边切割又考虑了负载平衡, 因此取得了较高的划分质量. 文献[98]是针对边划分提出的重划分算法, 具有较高的划分质量和较低的性能开销, 适合划分具有幂律分布的大规模的动态图数据. Aragon、Paragon 和 Planar 适用于在异构计算环境中的图划分问题;

- 在针对图计算动态的重划分算法中, 重点是要根据不同的图计算任务灵活处理, 这些算法主要适用于那些较高的负载平衡和较低的通信代价带来的收益远远大于顶点或边的重分配造成额外开销的图划分任务.

5 总结与展望

由于性能开销的影响, 图的动态性导致的划分质量下降的问题是传统静态图划分算法无法解决的, 因此, 近期有一些工作针对动态图划分问题进行了研究. 本节首先总结分析了不同类别的动态图划分算法的特点, 之后讨论了动态图划分问题存在的挑战, 并对未来的研究方向加以展望.

5.1 动态图划分算法的分类对比与分析

本文将动态图划分算法分为流式图划分算法、增量式图划分算法和图重划分算法这三大类, 详细的分类汇总可见表 4. 流式图划分算法依次处理顶点流或者边流, 可以用来划分动态图. 增量式图划分算法为动态变化部分ΔG提出了处理策略, 该算法的特点是更专注于处理和划分ΔG, 通过将ΔG的划分与原图 G 的划分相融合, 从而获得动态图 G'的划分. 而图重划分算法的特点是没有单独的划分策略来处理ΔG, 它假设图的动态变化已经导致分区发生了变化, 更专注于通过局部调整不断地优化分区, 从而获得高质量的划分结果.

表 4 动态图划分算法总结

类型	子类	划分方式	特点
流式图划分算法	基于哈希的划分算法	边割[27] 点割[23,24] 混合割[12]	使用一致的哈希函数将顶点或边映射到不同的分区中. 效率高, 具有较低的延迟, 但划分质量低
	基于邻居分布的划分算法	边割[27–29,79] 点割[11,32] 混合割[12]	利用新到达的顶点和边的邻居信息, 在映射函数中既考虑顶点和边的连接情况, 也考虑分区负载大小. 划分质量较高
	基于流的优化划分算法	边割[30,31,80] 点割[33,34,81–83] 混合割[84]	利用提前获取图信息、局部优化和分布式流这 3 种策略解决无法得知后续流信息的单元素流算法(哈希或者邻居分布)存在的弊端. 划分质量高, 但效率较低
增量式图划分算法	单元素增量式划分算法	边割[85] 点割[86,87] 混合割[88,89]	为每个单独的动态变化(单个顶点或边的插入或删除)提供划分策略. 实时处理持续增长图, 具有较好的划分质量, 较低开销
	批处理增量式划分算法	边割[26,90] 点割[26]	针对一批动态变化的划分策略, 能够保证一定的划分质量, 但是不具有实时性, 有一些算法不能够处理大规模图且开销大
图重划分算法	针对结构动态的划分算法	边割[35,91,92,94–97] 点割[98]	专注于在图结构发生动态变化后对分区进行优化, 通过分区间顶点或边的转移来提高图划分的质量
	针对计算动态的划分算法	边割[36,37,99–101] 点割[102]	专注于在真实的图计算应用中, 被激活的顶点的规模发生动态变化后通过顶点或边的转移对分区进行优化

流式图划分算法支持动态地向分区中添加顶点和边, 可以处理动态图中拓扑结构的变化. 最近的一些关于流式图划分的研究逐渐跳出单元素流和分区结构无法调整的设定, 获得了比较好的效果. 但是流式动态图划分算法只能处理顶点和边的插入, 而不能很好地处理顶点和边的删除. 一些研究者提出了增量式图划分算法, 分为单元素增量式图划分和批量增量式图划分: 单元素增量式图划分的研究很多都采用了重分配的思想; 批量增量式图划分研究中, 一个很重要的工作是利用已有的静态图划分算法得到动态图划分算法. 图的重划分算法分为针对图结构动态的算法和针对图计算动态的算法. 图的重划分是提升划分质量的主要手段, 大部

分动态图划分算法的研究都采用了重划分的思想, 通过迭代进行顶点或边的转移, 逐渐减少分区间的交互量.

基于流的方法是加载图的一种很好的选择, 因为它们不需要将整个图放在内存中进行划分. 顶点或边流可以同时在管道中加载和划分. 因此, 一些分布式图处理系统将流方法作为默认的划分方法. 此外, 流式图划分算法中的基于哈希和基于邻居分布的算法由于具有较高的效率, 一般用于图的初始划分. 增量式动态图划分算法中的单元素增量式算法适合用于实时场景中, 能够获得更好的划分质量和性能. 针对结构动态的重划分算法能够处理由图发生动态变化而导致的划分质量下降的问题, 拥有较好的划分质量. 而针对计算动态的重划分算法能够切实考虑到真实的图计算和处理的运行过程. 在迭代计算中, 针对不同的图计算任务的特点, 通过顶点或者边的转移不断提高分布式图处理的性能.

5.2 动态图划分的挑战

本文对动态图划分算法进行了分析、总结和归纳. 动态图划分虽然已经有了比较深入的研究, 取得了一定的成果, 但仍面临不少困难和挑战, 主要如下.

1) 基于边割的动态图划分已经有较为完善的研究. 大量的研究证明: 基于点割的图划分在处理具有幂律分布特点的真实图方面具有更好的效果, 而对基于点割的动态图划分的研究还并不多, 尤其是基于点割的图重划分算法, 其中的难点在于如何找到能够减少交互量的边进行转移;
2) 大多数图重划分算法并不关心图是如何发生动态变化的, 实际上, 它们只是对动态图的一个快照通过局部调整进行优化, 比较适合离线场景. 然而, 真实图是实时动态变化的, 如何在满足实时性的条件下获得高质量的划分结果, 是动态图划分的一大挑战;
3) 对于具有幂律分布特点的真实图, 研究表明: 基于边割的划分有着更好的局部性, 而基于点割的划分有着更好的负载平衡. 混合割结合了边割和点割的优势. 然而, 现有的工作对混合割的研究较少, 仅仅在流式图划分算法和增量式图划分算法中有少量的工作. 因此, 研究高性能的基于混合割的动态图划分方法是一大挑战;
4) 目前, 大多数图划分算法都是基于简单图, 而从真实世界中抽象出来的图是复杂的, 比如有向图、带权图、数据流图、概率图、复杂网络、超图以及知识图谱等. 如何设计合理的图划分算法处理特殊图也是一大挑战;
5) 在如今的大数据时代, 从真实应用中抽象出来的图数据是超大规模和动态变化的. 集中式环境下的图划分算法已经难以适应当前应用的需求, 分布式并行环境下的图划分算法的研究日益迫切;
6) 目前的研究大多都以最小化分区间的交互量和平衡分区负载为目标, 而现实的图处理场景非常复杂. 由于计算环境的异构性、图计算的动态性等原因, 最小化交互量和平衡分区负载并不等同于最大化图计算的性能. 图划分的最终目的是提升图计算的性能, 因此, 如何将图划分与真实的分布式图处理结合是一大挑战.

5.3 未来的研究方向

综上所述, 从当前的研究现状来看, 动态图划分问题的研究还有继续开展的空间. 随着数据规模的快速扩张, 如何在实时场景中以分布式的方式处理动态图的划分, 将会是研究的一个主要方向. 如何考虑计算结构异构性和计算动态性, 更好地结合图划分和真实的大规模分布式图处理, 以及针对一些特殊图设计动态图划分方法等, 是研究的一大重点. 除此之外, 基于混合割的图划分由于它在真实图上的划分效果更好, 也将逐渐成为研究的主流. 未来的研究方向主要包括以下几个方面.

1) 基于点割的动态图重划分算法的研究. 现有的动态重划分方法大多基于边割, 然而, 许多分布式框架使用点割模型. 因此, 研究高性能的基于点割的动态图重划分算法, 是未来的一大研究方向. 此外, 现有的动态重划分方法是针对同步系统的. 在异步系统中, 顶点或边的迁移更为复杂, 这需要锁等并发控制机制来避免不一致. 如何设计异步的动态重划分方法, 也是未来的一个研究方向.
2) 基于混合割的动态图划分算法的研究. 基于混合割的划分策略能够结合边割和点割各自的优势. 对

于具有幂律分布特点的真实图, 基于混合割的划分方法实现了更好的分区负载平衡和最小化分区间的通信量. 因此, 研究高性能的基于混合割的动态图划分算法, 将会是未来的一大研究方向.

3) 实时动态图划分算法的研究. 已有的大量工作都是对动态图的一个快照进行分析, 并没有提出实时处理动态变化的策略. 而如今, 有一些图应用需要实时处理动态变化图, 以满足高性能的分布式图处理任务. 因此, 如何研究单元素增量式动态图划分算法, 能够满足实时性并获得高质量划分结果, 将是未来的一个研究方向.

4) 针对特殊图的动态图划分算法的研究. 从真实应用中抽象出来的图数据结构是千变万化的, 如有向图、带权图等. 此外, 还有一些富有特殊含义的图, 如数据流图、概率图、复杂网络等. 如何研究出针对这些特殊图的分布式并行环境下的动态图划分算法, 也是未来的一个研究方向.

5) 针对知识图谱和超图的动态图划分算法的研究. 超图是图的一种扩展形式, 其中一条边(通常称为超边或网)可以连接任意数量的顶点. 知识图谱也是图的某种扩展形式, 其在顶点和边上往往会附加更多的属性信息, 用于描述现实世界中事物的广泛联系. 虽然已有一些工作针对知识图谱和超图提出了划分算法, 但大多是基于静态图的. 因此, 研究高性能的动态图划分算法划分超大规模知识图谱和超图, 是未来的一大研究方向.

6) 针对硬件异构性的动态图划分算法的研究. 在真实的分布式图处理中, 处于不同地理位置的机器的硬件资源存在异构性, 如 CPU、GPU、存储容量等, 并且不同机器间通信时的网络带宽和网络价格不一致. 因此, 如何设计针对计算异构性的动态图划分算法, 也是未来的一大研究方向.

7) 针对图计算动态性的动态图划分算法的研究. 对于不同的分布式图处理任务, 有着不同的通信和计算模式, 故即使是以最小化分区间的交互量和平衡分区负载为目标的最优的图划分方法, 也不能保证所有图计算任务的性能. 这就需要考虑图计算运行时特性, 更好地将图划分与真实的大规模分布式图处理相结合. 因此, 如何利用机器学习、深度学习、图论、统计学等技术设计任务驱动的动态图划分算法, 将会是未来的一大研究方向.

References:

[1] Stanford large network dataset collection. http://snap.stanford.edu/data/index.html

[2] Konect. http://konect.cc/networks/

[3] Network repository. http://networkrepository.com/networks.php

[4] Laboratory for Web algorithmics. http://law.di.unimi.it/datasets.php

[5] Leskovec J, Dumais S, Horvitz E. Web projections: Learning from contextual subgraphs of the Web. In: Proc. of the 16th Int'l Conf. on World Wide Web (WWW). ACM, 2007. 471−480.

[6] Chakrabarti D, Zhan Y, Faloutsos C. R-MAT: A recursive model for graph mining. In: Proc. of the 4th SIAM Int'l Conf. on Data Mining (SDM). 2004. 442−446.

[7] Leskovec J, Chakrabarti D, Kleinberg JM, Faloutsos C, Ghahramani Z. Kronecker graphs: An approach to modeling networks. Journal of Machine Learning Research, 2010, 11(Feb.): 985−1042.

[8] Malewicz G, Austern MH, Bik AJ, Dehnert J, Horn I, Leiser N, Czajkowski G. Pregel: A system for large-scale graph processing. In: Proc. of the 2010 Int'l Conf. on Management of Data (SIGMOD). 2010. 135−146.

[9] Giraph. https://giraph.apache.org/

[10] Low Y, Bickson D, Gonzalez J, Guestrin C, Kyrola A, Hellerstein JM. Distributed GraphLab: A framework for machine learning and data mining in the cloud. Proc. of the VLDB Endowment, 2012, 5(8): 716−727.

[11] Gonzalez JE, Low Y, Gu H, Bickson D, Guestrin C. Powergraph: Distributed graph-parallel computation on natural graphs. In: Proc. of the 10th USENIX Symp. on Operating Systems Design and Implementation (OSDI). 2012. 17−30.

[12] Chen R, Shi J, Chen Y, Zang B, Guan H, Chen H. Powerlyra: Differentiated graph computation and partitioning on skewed graphs. ACM Trans. on Parallel Computing (TOPC), 2018, 5(3): 13:1−13:39.

[13] Gonzalez JE, Xin RS, Dave A, Crankshaw D, Franklin MJ, Stoica I. Graphx: Graph processing in a distributed dataflow framework. In: Proc. of the 11th USENIX Symp. on Operating Systems Design and Implementation (OSDI). 2014. 599−613.

[14] Li D, Zhang Y, Wang J, Tan K. TopoX: Topology refactorization for efficient graph partitioning and processing. Proc. of the VLDB Endowment, 2019, 12(8): 891−905.

[15] Cui PJ. Yuan Y, Li CH, Zhang C, Wang RG. RGraph: An effective distributed graph processing system based on RDMA. Ruan Jian Xue Bao/Journal of Software, 2022, 33(3): 1018−1042 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6449.htm [doi: 10.13328/j.cnki.jos.006449]

[16] Donath WE, Hoffman AJ. Algorithms for partitioning of graphs and computer logic based on eigenvectors of connection matrices. IBM Technical Disclosure Bulletin, 1972, 15(3): 938−944.

[17] Donath WE, Hoffman AJ. Lower bounds for the partitioning of graphs. IBM Journal of Research and Development, 1973, 17(5): 420−425.

[18] Simon HD. Partitioning of unstructured problems for parallel processing. Computer Systems Science and Engineering, 1991, 2(2): 135−148.

[19] Williams RD. Performance of dynamic load balancing algorithms for unstructured mesh calculations. Concurrency and Computation: Practice and Experience, 1991, 3(5): 457−481.

[20] Karypis G. METIS: Unstructured graph partitioning and sparse matrix ordering system. Technical Report, 1997.

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

[22] Karypis G, Kumar V. Multilevel graph partitioning schemes. In: Proc. of the 1995 Int'l Conf. on Parallel Processing (ICPP). 1995. 113−122.

[23] Xie C, Yan L, Li WJ, Zhang Z. Distributed power-law graph computing: Theoretical and empirical analysis. In: Advances in Neural Information Processing Systems (NIPS). 2014. 1673−1681.

[24] Jain N, Liao G, Willke TL. Graphbuilder: Scalable graph ETL framework. In: Proc. of the 1st Int'l Workshop on Graph Data Management Experiences and Systems. ACM, 2013.

[25] Hanai M, Suzumura T, Tan WJ, Liu ES, Theodoropoulos G, Cai W. Distributed edge partitioning for trillion-edge graphs. Proc. of the VLDB Endowment, 2019, 12(13): 2379−2392.

[26] Fan W, Liu M, Tian C, Xu R, Zhou J. Incrementalization of graph partitioning algorithms. Proc. of the VLDB Endowment, 2020, 13(8): 1261−1274.

[27] Stanton I, Kliot G. Streaming graph partitioning for large distributed graphs. In: Proc. of the 18th Int'l Conf. on Knowledge Discovery and Data Mining (SIGKDD). ACM, 2012. 1222−1230.

[28] Tsourakakis CE, Gkantsidis C, Radunovic B, Vojnovic M. Fennel: Streaming graph partitioning for massive scale graphs. In: Proc. of the 7th ACM Int'l Conf. on Web Search and Data Mining (WSDM). ACM, 2014. 333−342.

[29] Zhang W, Chen Y, Dai D. AKIN: A streaming graph partitioning algorithm for distributed graph storage systems. In: Proc. of the 18th IEEE/ACM Int'l Symp. on Cluster, Cloud and Grid Computing. IEEE, 2018. 183−192.

[30] Patwary MAK, Garg S, Kang B. Window-based streaming graph partitioning algorithm. In: Proc. of the Australasian Computer Science Week Multiconference. ACM, 2019. 1−10.

[31] Firth H, Missier P, Aiston J. Loom: Query-aware partitioning of online graphs. In: Proc. of the 21st Int'l Conf. on Extending Database Technology (EDBT). 2018. 337−348.

[32] Petroni F, Querzoni L, Daudjee K, Kamali S, Iacoboni G. Hdrf: Stream-based partitioning for power-law graphs. In: Proc. of the 24th Int'l Conf. on Information and Knowledge Management (CIKM). ACM, 2015. 243−252.

[33] Mayer C, Mayer R, Tariq MA, Geppert H, Laich L, Rieger L, Rothermel K. Adwise: Adaptive window-based streaming edge partitioning for high-speed graph processing. In: Proc. of the 38th Int'l Conf. on Distributed Computing Systems (ICDCS). IEEE, 2018. 685−695.

[34] Sajjad HP, Payberah AH, Rahimian F, Vlassov V, Haridi S. Boosting vertex-cut partitioning for streaming graphs. In: Proc. of the 2016 IEEE Int'l Congress on Big Data (BigData Congress). IEEE, 2016. 1−8.

[35] Nicoara D, Kamali S, Daudjee K, Chen L. Hermes: Dynamic partitioning for distributed social network graph databases. In: Proc. of the 18th Int'l Conf. on Extending Database Technology (EDBT). 2015. 25−36.

[36] Mayer C, Tariq MA, Mayer R, Rothermel K. Graph: Traffic-aware graph processing. IEEE Trans. on Parallel and Distributed Systems, 2018, 29(6): 1289−1302.

[37] Xu N, Chen L, Cui B. LogGP: A log-based dynamic graph partitioning method. Proc. of the VLDB Endowment, 2014, 7(14): 1917−1928.

[38] Gill G, Dathathri R, Hoang L, Pingali K. A study of partitioning policies for graph analytics on large-scale distributed platforms. Proc. of the VLDB Endowment, 2018, 12(4): 321−334.

[39] Adoni HWY, Nahhal T, Krichen M, Aghezzaf B, Elbyed A. A survey of current challenges in partitioning and processing of graph-structured data in parallel and distributed systems. Distributed and Parallel Databases, 2020, 38(2): 495−530.

[40] Pacaci A, Özsu MT. Experimental analysis of streaming algorithms for graph partitioning. In: Proc. of the 2019 Int'l Conf. on Management of Data (SIGMOD). ACM, 2019. 1375−1392.

[41] Abbas Z, Kalavri V, Carbone P, Vlassov V. Streaming graph partitioning: An experimental study. Proc. of the VLDB Endowment, 2018, 11(11): 1590−1603.

[42] Verma S, Leslie LM, Shin Y, Gupta I. An experimental comparison of partitioning strategies in distributed graph processing. Proc. of the VLDB Endowment, 2017, 10(5): 493−504.

[43] Soudani NM, Fatemi A, Nematbakhsh M. An investigation of big graph partitioning methods for distribution of graphs in vertex-centric systems. Distributed and Parallel Databases, 2020, 38(1): 1−29.

[44] Xu JF, Dong YH, Wang SY, He XM, Chen HH. A review of algorithms for partitioning large-scale graph data. Dian Xin Ke Xue/ Telecommunications Science, 2014, 30(7): 100−106 (in Chinese with English abstract). [doi: 10.3969/j.issn.1000-0801.2014.07.016]

[45] Karypis G, Aggarwal R, Kumar V, Shekhar S. Multilevel hypergraph partitioning: Applications in VLSI domain. IEEE Trans. on Very Large Scale Integration (VLSI) Systems, 1999, 7(1): 69−79.

[46] Alpert CL, Hagen LW, Kahng AB. A hybrid multilevel/genetic approach for circuit partitioning. In: Proc. of the Asia Pacific Conf. on Circuits and Systems (APCCAS'96). IEEE, 1996.

[47] Zhang E, Gao L. A circuit partition algorithm based on point cut. Chinese Journal of Computers, 2014, 37(7): 1528−1537 (in Chinese with English abstract). [doi: 10.3724/SP.J.1016.2014.01528]

[48] Pellegrini F, Roman J. Scotch: A software package for static mapping by dual recursive bipartitioning of process and architecture graphs. In: Proc. of the High-Performance Computing and Networking. Springer-Verlag, 1996. 493−498.

[49] Hendrickson B, Leland RW. A multi-level algorithm for partitioning graphs. In: Proc. of the SC'95 Conf. ACM, 1995.

[50] Deng L, Dai N, Xu B, Xing C, Chen M. PNFVNbM: A method to partition large-scale NFV networks based on Metis. Chinese Journal of Computers, 2020, 43(10): 1958−1968 (in Chinese with English abstract). [doi: 10.11897/SP.J.1016.2020.01958]

[51] Karypis G, Kumar V. Parallel multilevel series *k*-way partitioning scheme for irregular graphs. SIAM Review, 1999, 41(2): 278−300.

[52] Chevalier C, Pellegrini F. PT-scotch: A tool for efficient parallel graph ordering. Parallel Computing, 2008, 34(6−8): 318−331.

[53] Rahimian F, Payberah AH, Girdzijauskas S, Jelasity M, Haridi S. Ja-be-Ja: A distributed algorithm for balanced graph partitioning. In: Proc. of the 7th Int'l Conf. on Self-adaptive and Self-Organizing Systems (SASO). IEEE, 2013. 51−60.

[54] Metis. http://glaros.dtc.umn.edu/gkhome/views/metis

[55] Kernighan BW, Lin S. An efficient heuristic procedure for partitioning graphs. Bell System Technical Journal, 1970, 49(2): 291−307.

[56] Fiduccia CM, Mattheyses RM. A linear-time heuristic for improving network partitions. In: Proc. of the 19th Design Automation Conf. (DAC). IEEE, 1982. 175−181.

[57] LaSalle D, Karypis G. A parallel hill-climbing refinement algorithm for graph partitioning. In: Proc. of the 45th Int'l Conf. on Parallel Processing (ICPP). IEEE, 2016. 236−241.

[58] Xu JF, Dong YH, Wang SY, He XM, Chen HH. LGP-SA: Large-scale graph partitioning algorithm based on simulated annealing in distributed environment. Dian Xin Ke Xue/Telecommunications Science, 2016, 32(2): 83–91 (in Chinese with English abstract). [doi: 10.11959/j.issn.1000-0801.2016078]

[59] Wang L, Xiao Y, Shao B, Wang H. How to partition a billion-node graph. In: Proc. of the 30th Int'l Conf. on Data Engineering (ICDE). IEEE, 2014. 568−579.

[60] Martella C, Logothetis D, Loukas A, Siganos G. Spinner: Scalable graph partitioning in the cloud. In: Proc. of the 33rd Int'l Conf. on Data Engineering (ICDE). IEEE, 2017. 1083−1094.

[61] Yan J, Tan G, Sun N. Study on partitioning real-world directed graphs of skewed degree distribution. In: Proc. of the 44th Int'l Conf. on Parallel Processing (ICPP). IEEE, 2015. 699−708.

[62] Zheng A, Labrinidis A, Faloutsos C. Skew-resistant graph partitioning. In: Proc. of the 33rd Int'l Conf. on Data Engineering (ICDE). IEEE, 2017. 151−154.

[63] Kim M, Candan KS. SBV-cut: Vertex-cut based graph partitioning using structural balance vertices. Data & Knowledge Engineering, 2012, 72: 285−303.

[64] Rahimian F, Payberah AH, Girdzijauskas S, Haridi S. Distributed vertex-cut partitioning. In: Proc. of the Distributed Applications and Interoperable Systems (DAIS). Springer, 2014. 186−200.

[65] Zhang C, Wei F, Liu Q, Tang ZG, Li Z. Graph edge partitioning via neighborhood heuristic. In: Proc. of the 23rd Int'l Conf. on Knowledge Discovery and Data Mining (SIGKDD). ACM, 2017. 605−614.

[66] Ji S, Bu C, Li L, Wu X. Local graph edge partitioning with a two-stage heuristic method. In: Proc. of the 39th IEEE Int'l Conf. on Distributed Computing Systems (ICDCS). IEEE, 2019. 228−237.

[67] Zhang Y, Liu Y, Yu J, Liu P, Guo L. Vsep: A distributed algorithm for graph edge partitioning. In: Proc. of the Int'l Conf. on Algorithms and Architectures for Parallel Processing. Springer, 2015. 71−84.

[68] Li Y, Constantin C, Mouza C. SGVcut: A vertex-cut partitioning tool for random walks-based computations over social network graphs. In: Proc. of the 29th Int'l Conf. on Scientific and Statistical Database Management. ACM, 2017.

[69] Margo D, Seltzer M. A scalable distributed graph partitioner. Proc. of the VLDB Endowment, 2015, 8(12): 1478−1489.

[70] Guerrieri A, Montresor A. DFEP: Distributed funding-based edge partitioning. In: Proc. of the European Conf. on Parallel Processing. Springer, 2015. 346−358.

[71] Li D, Zhang Y, Wang J, Tan KL. TopoX: Topology refactorization for efficient graph partitioning and processing. Proc. of the VLDB Endowment, 2019, 12(8): 891−905.

[72] Zhu X, Chen W, Zheng W, Ma X. Gemini: A computation-centric distributed graph processing system. In: Proc. of the OSDI 2016. USENIX Association, 2016. 301−316.

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

[74] Fan W, Jin R, Liu M, Lu P, Luo X, Xu R, Yin Q, Yu W, Zhou J. Application driven graph partitioning. In: Proc. of the SIGMOD. ACM, 2020. 1765−1779.

[75] Wang Z, Gu Y, Bao Y, Yu G. OnFlyP: Distributed online large graph partitioning algorithm based on directional edge switching. Chinese Journal of Computers, 2015, 38(9): 1838−1851 (in Chinese with English abstract). [doi: 10.11897/SP.J.1016.2015.01838]

[76] Wang X, Chen W, Yang Y, Zhang X, Feng Z. A review of knowledge graph partitioning algorithms. Chinese Journal of Computers, 2021, 44(1): 235−260 (in Chinese with English abstract). [doi: 10.11897/SP.J.1016.2021.00235]

[77] Gottesbüren L, Heuer T, Sanders P, Schlag S. Scalable shared-memory hypergraph partitioning. In: Proc. of the Workshop on Algorithm Engineering and Experiments (ALENEX). 2021. 16−30.

[78] Papa DA, Markov I. Hypergraph partitioning and clustering. In: Handbook of Approximation Algorithms and Metaheuristics. 2007.

[79] LI Q, LI H, Zhong J, Ying CT, LI Q. Research on graph partitioning in heterogeneous computing environment. Chinese Journal of Computers, 2021, 44(8): 1751−1766 (in Chinese with English abstract). [doi: 10.11897/SP.J.1016.2021.01751]

[80] Nishimura J, Ugander J. Restreaming graph partitioning: Simple versatile algorithms for advanced balancing. In: Proc. of the 19th Int'l Conf. on Knowledge Discovery and Data Mining (SIGKDD). ACM, 2013. 1106−1114.

[81] Mayer R, Orujzade K, Jacobsen HA. 2PS: High-quality edge partitioning with two-phase streaming. arXiv:2001. 07086, 2020.

[82] Taimouri M, Saadatfar H. RBSEP: A reassignment and buffer based streaming edge partitioning approach. Journal of Big Data, 2019, 6(1): 1−17.

[83] Hua QS, Li Y, Yu D, Jin H. Quasi-streaming graph partitioning: A game theoretical approach. IEEE Trans. on Parallel and Distributed Systems, 2019, 30(7): 1643−1656.

[84] Davoudian A, Chen L, Tu H, Liu M. A workload-adaptive streaming partitioner for distributed graph stores. Data Science and Engineering, 2021, 6(2): 163−179.

[85] Abdolrashidi A, Ramaswamy L. Continual and cost-effective partitioning of dynamic graphs for optimizing big graph processing systems. In: Proc. of the 2016 IEEE Int'l Congress on Big Data (BigData Congress). IEEE, 2016. 18−25.

[86] Li H, Yuan H, Huang J, Cui J, Ma X, Wang S, Yoo J, Yu PS. Group reassignment for dynamic edge partitioning. IEEE Trans. on Parallel and Distributed Systems, 2021, 32(10): 2477−2490.

[87] Sakouhi C, Aridhi S, Guerrieri A, Sassi S, Montresor A. Dynamicdfep: A distributed edge partitioning approach for large dynamic graphs. In: Proc. of the 20th Int'l Database Engineering & Applications Symposium (IDEAS). ACM, 2016. 142−147.

[88] Dai D, Zhang W, Chen Y. IOGP: An incremental online graph partitioning algorithm for distributed graph databases. In: Proc. of the 26th Int'l Symp. on High-performance Parallel and Distributed Computing. ACM, 2017. 219−230.

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

[90] Ou CW, Ranka S. Parallel incremental graph partitioning using linear programming. In: Proc. of the 1994 Conf. on Supercomputing. IEEE, 1994. 458−467.

[91] Schloegel K, Karypis G, Kumar V. Dynamic repartitioning of adaptively refined meshes. In: Proc. of the 1998 ACM/IEEE Conf. on Supercomputing. ACM, 1998.

[92] Vaquero LM, Cuadrado F, Logothetis D, Martella C. Adaptive partitioning for large-scale dynamic graphs. In: Proc. of the 34th Int'l Conf. on Distributed Computing Systems (ICDCS). IEEE, 2014. 144−153.

[93] Neo4j. https://neo4j.com/

[94] Li H, Yuan H, Huang J, Cui J, Yoo J. Dynamic graph repartitioning: From single vertex to vertex group. In: Proc. of the 25th Int'l Conf. on Database Systems for Advanced Applications (DASFAA). Springer, 2020. 482−497.

[95] Zheng A, Labrinidis A, Chrysanthis PK. Architecture-aware graph repartitioning for data-intensive scientific computing. In: Proc. of the 2014 IEEE Int'l Conf. on Big Data (Big Data). IEEE, 2014. 78−85.

[96] Zheng A, Labrinidis A, Pisciuneri PH, Chrysanthis PK, Givi P. PARAGON: Parallel architecture-aware graph partition refinement algorithm. In: Proc. of the 19th Int'l Conf. on Extending Database Technology (EDBT). 2016. 365−376.

[97] Zheng A, Labrinidis A, Chrysanthis PK. Planar: Parallel lightweight architecture-aware adaptive graph repartitioning. In: Proc. of the 32nd Int'l Conf. on Data Engineering (ICDE). IEEE, 2016. 121−132.

[98] Li H, Yuan H, Huang J. Real-time edge repartitioning for dynamic graph. In: Proc. of the 28th ACM Int'l Conf. on Information and Knowledge Management (CIKM). ACM, 2019. 2125−2128.

[99] Salihoglu S, Widom J. GPS: A graph processing system. In: Proc. of the Scientific and Statistical Database Management (SSDBM). 2013. 1−12.

[100] Khayyat Z, Awara K, Alonazi A, Jamjoom H, Williams D, Kalnis P. Mizan: A system for dynamic load balancing in large-scale graph processing. In: Proc. of the 8th European Conf. on Computer Systems (EuroSys). ACM, 2013. 169−182.

[101] Shang Z, Yu JX. Catch the wind: Graph workload balancing on cloud. In: Proc. of the 29th Int'l Conf. on Data Engineering (ICDE). IEEE, 2013. 553−564.

[102] Kumar D, Raj A, Dharanipragada J. GraphSteal: Dynamic re-partitioning for efficient graph processing in heterogeneous clusters. In: Proc. of the 10th Int'l Conf. on Cloud Computing (CLOUD). IEEE, 2017, 439−446.

附中文参考文献:

[15] 崔鹏杰, 袁野, 李岑浩, 张灿, 王国仁. RGraph: 基于 RDMA 的高效分布式大图数据处理系统. 软件学报, 2022, 33(3): 1018−1042. http://www.jos.org.cn/1000-9825/6449.htm [doi: 10.13328/j.cnki.jos.006449]

[44] 许金凤, 董一鸿, 王诗懿, 何贤芒, 陈华辉. 大规模图数据划分算法综述. 电信科学, 2014, 30(7): 100–106. [doi: 10.3969/j.issn.1000-0801.2014.07.016]

[47] 张恩利, 高琳. 一种基于点割的电路划分算法. 计算机学报, 2014, 37(7): 1528–1537. [doi: 10.3724/SP.J.1016.2014.01528]

[50] 邓理, 戴宁赟, 许博, 邢长友, 陈鸣. PNFVNbM: 一种基于 Metis 划分大规模 NFV 网络的方法. 计算机学报, 2020, 43(10): 1958–1968. [doi: 10.11897/SP.J.1016.2020.01958]

[58] 许金凤, 董一鸿, 王诗懿, 何贤芒, 陈华辉. LGP-SA: 分布式环境下基于模拟退火的大规模图划分算法. 电信科学, 2016, 32(2): 83–91. [doi: 10.11959/j.issn.1000-0801.2016078]

[75] 王志刚, 谷峪, 鲍玉斌, 于戈. OnFlyP: 基于定向边交换的分布式在线大图划分算法. 计算机学报, 2015, 38(9): 1838–1851. [doi: 10.11897/SP.J.1016.2015.01838]

[76] 王鑫, 陈蔚雪, 杨雅君, 张小旺, 冯志勇. 知识图谱划分算法研究综述. 计算机学报, 2021, 44(1): 235–260. [doi: 10.11897/SP.J.1016.2021.00235]

[79] 李琪, 李虎雄, 钟将, 英昌甜, 李青. 异构计算环境中图划分算法的研究. 计算机学报, 2021, 44(8): 1751–1766. [doi: 10.11897/SP.J.1016.2021.01751]

李贺(1983－), 男, 博士, 副教授, CCF 专业会员, 主要研究领域为图数据管理, 图数据挖掘, 深度学习.

刘延娜(1996－), 女, 硕士, 主要研究领域为大图分割, 分布式计算, 图数据挖掘.

袁航(1995－), 男, 硕士, 主要研究领域为大图分割, 分布式计算.

杨舒琪(2001－), 女, 硕士生, 主要研究领域为大图分割, 分布式计算, 深度学习.

韵晋鹏(1996－), 男, 硕士生, 主要研究领域为异常检测, 时间序列预测.

乔少杰(1981－), 男, 博士, 教授, CCF 杰出会员, 主要研究领域为数据库, 人工智能, 数据挖掘.

黄健斌(1975－), 男, 博士, 教授, CCF 专业会员, 主要研究领域为数据挖掘, 智慧交通, 人工智能.

崔江涛(1976－), 男, 博士, 教授, CCF 杰出会员, 主要研究领域为数据库, 大数据, 区块链.