图分区算法综述

廖宇健[1)]

[1)](华中科技大学计算机科学与技术学院, 武汉 430074)

摘　要　图划分对于大型图并行计算来说至关重要。在单一的机器上，内存缓存磁盘读写，处理器数量，显卡数量都有一个上限，对于大型图来说，很容易出现上限问题，由此诞生了图划分。图划分问题，就是将一个大型图划分为若干个子图以便在分布式系统中运行。图划分的优化目标包括两项：负载均衡和最小割 (cut)。二者都是为了提高在分布式系统中运算的性能。负载均衡是为了使分布式系统中的多台计算机有相近的任务负荷，避免少数计算机负载过高。最小割则是为了减少计算机之间的通信代价。同时优化两个目标目前已知是 NP 完全问题。图划分算法是实现并行资源高效分配的关键技术。由于图数据固有的高连通性和强耦合性，在对图数据进行并行计算时，需要尽可能的降低子图之间的耦合度，同时提高子图内部的连通性。而图划分技术是实现解耦的重要手段。为了处理现实生活中的图，图分区通常是必须的。它是将一个大图 G 切割成更小的片段，并将这些片段分配给一组处理器（也称为工作线程），使工作线程有均匀的并行计算工作量，并最大限度地减少它们之间的通信。图划分算法已经成为主流，通常可以分为 edge-cut 和 vertex-cut，edge-cut 顾名思义为切分边，均匀的划分顶点，或者 vertex-cut，通过复制顶点来均匀的划分边，还有混合分区器，同时切割边和顶点。这些分区器通常遵循两个质量标准，平衡和复制因子。为了平衡工作负载并减少同步开销，分区器通常试图将图切割成“均匀”大小的片段，并减少复制的边和顶点。然而，在现实世界中，这些标准并不总是能捕捉到影响并行图算法性能的瓶颈因素，因为算法的计算和通信模式各不相同。因此，近期的图分区算法针对特定的算法，捕捉内在的评价指标，进行图分区优化。

关键词　图分区；大型图；经典图算法；负载均衡；通信负载

中图法分类号　TP391　　DOI 号　10.11897/SP.J.1016.01.2022.00001

Overview of Graph Partitioning

Yujian Liao[1)]

[1)](School of Science and Technology, Huazhong University of Science and Technology, Wuhan, 430074, China)

Abstract　Graph partitioning is essential for parallel computing of large graphs. On a single machine, there is an upper limit for the number of memory, cache, disk reads and writes, the number of processors, and the number of graphics cards. For large graphs, the upper limit problem is prone to occur, which gives birth to graph division. The problem of graph division is to divide a large graph into several subgraphs to run in a distributed system. The optimization goals of graph division include two: load balancing and minimum cut. Both are to improve the performance of computing in distributed systems. Load balancing is to make multiple computers in a distributed system have similar task loads, and to avoid excessive load on a small number of computers. The minimum cut is to reduce the communication cost between computers. Optimizing two goals at the same time is currently known as an NP complete problem. The graph partition algorithm is a key technology to realize the efficient allocation of parallel resources. Due to the inherent high connectivity and strong coupling of graph data, when performing parallel calculations on graph data, it is necessary to reduce the coupling between subgraphs as much as possible while improving the connectivity within the subgraphs. The graph partition technology is an important means to achieve decoupling. In order to deal with real-life graphs, graph partitioning is usually necessary. It cuts a large graph G into smaller fragments, and allocates these fragments to a group of processors (also called worker threads), so that the worker threads have uniform parallel computing workload and minimize them. Inter-communication. Graph partitioning algorithms have become mainstream, and they can usually be divided into edge-cut and vertex-cut. Edge-cut, as the name suggests, means splitting edges, dividing vertices evenly, or vertex-cut, dividing edges evenly by copying vertices, and mixing Partitioner, cutting edges and vertices at the same time. These partitioners usually follow two quality standards, balance and replication factor. In order to balance the workload and reduce synchronization overhead, partitioners usually try to cut the graph into ”uniform” sized fragments, and reduce the copied edges and vertices. However, in the real world, these standards do not always capture the bottleneck factors that affect the performance of parallel graph algorithms, because the calculation and communication modes of the algorithms are different. Therefore, the recent graph partitioning algorithm captures the inherent evaluation index for a specific algorithm, and optimizes the graph partition.

Key words　graph partition; large graph; classical graph algorithm; load balancing; communication load

1 引言

1.1 背景

图划分是指将一个图数据按照某种规则划分成多个子区 (点划分或者边划分)，其次需要考虑切割边或者点最少。因为同时优化这两个目标是 NP 完全问题，目前仍未发现有效的求解精确算法。现在主要有，谱图划分，几何划分，启发式搜索，多级优化，流式划分以及动态重构。

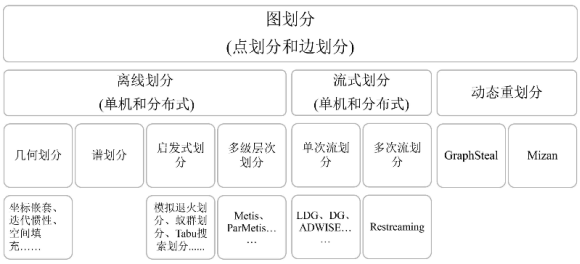


图 1 图划分分类

- 几何划分[1-5] 的核心思想是根据图的几何信息来切分图。不同的方案是利用顶点坐标信息寻找数据依赖关系的方法不同。由于几何方法只是简单的利用坐标信息而没有考虑到顶点的结构信息与顶点的度。所以该方法的划分质量较差，且计算顶点坐标信息的开销也较大。

- 谱方法[6-9] 主要是针对图的二划分而言的。它的基本思想是用图矩阵的第二大特征值和特征向量来实现图的划分。谱方法能为许多不同类的问题提供较好的划分，但是谱方法的计算量非常大。

- 启发式搜索[10-13] 主要是基于启发式规则的求解策略。主要思想是先将图随机划分，再交换任意两个节点对其收益值进行估价，再高效地从中选择收益高的点进行交换。

- 多层次划分算法[14-17] 主要包括三个阶段，粗粒度划分，初始化划分，反粗粒度划分。第一阶段通过粗糙化技术将大图归约到可接受的小图；第二阶段将第一阶段获得的小图进行随机划分，并进行优化；第三阶段通过反粗粒度优化技术将小图的划分还原为原图的划分。

- 流划分[18-20]，在面向分布式图计算的图划分中，流划分高效的划分管理得到了青睐。流式划分主要根据不同的启发式规则来分配当前的顶点。后面有人提出 restreaming 划分方法，经过多次的重复流划分，划分的质量明显提高。在此基础上提出了并行流划分用来提升流划分的效率。

- 分布式图处理系统的运行过程中，造成负载不均的因素有很多，包括初始图数据划分不均、活跃节点数目的变化等。为此，许多系统都设计了相应的动态重划分策略来保证负载均衡。使图处理过程中，系统能够动态保持分区负载的均衡。典型的使用动态重划分策略的系统有 GraphSteal[21]，Mizan[22] 等。

1.2 现有的进展

现有的算法已经提出了应用驱动的分区策略，不仅仅考虑传统的静态指标，还有大量的论文已经把动态图动态指标加入进去优化，比如应用驱动的图分区策略，就是加入了针对算法的动态指标和分区策略。

1.2.1 怎么样更好的评价图划分的结果

技术更新层出不穷，以云计算为代表的新兴技术突破了现有计算资源获取的屏障，用户随时随地都可以享受虚拟的云计算资源，但同时也造成计算环境的复杂多变，尤其是并行集群的使用，不可避免的会出现计算节点，网络带宽等各种不同性能硬件的异构。面对计算环境及实际应用需求的复杂多变，传统的以最小化割边数且均衡负载作为评价所有图划分结果的方式显然不能适用，其评价标准相对单一。如何去评价划分结果的优劣，是实现并行计算性能提升的关键。这些问题都还有待解决。

1.2.2 当前图划分算法存在什么问题

图划分问题的研究起步较早，从小图到大规模网络划分都有了大量的研究成果。但是，这些研究的局限性较大，大多数都无法适用于并行计算的图划分。当前的图划分算法存在怎样的问题？是我们研究的根本出发点。因为图划分问题本身是 NP 完全问题，至今还没有发现多项式时间复杂度的求解算法。

1.2.3 如何快速地寻找满足要求的图划分

图划分作为图论研究的重要内容之一有着极其丰富的内涵。从狭义上来说，图划分就是解决最小割问题，图划分的本质是为了提高并行计算的效

收稿日期：2022-01-05；修改日期：2022-01-05 本课题得到……基金中文完整名称 (No. 项目号)、……基金中文完整名称 (No. 项目号)、……基金中文完整名称 (No. 项目号) 资助. 廖宇健, E-mail: **********.

第 1 作者手机号码：……，E-mail: **********

率。随着互联网技术的普及，图查询的应用需求逐渐变得强烈，如交通路线的查询，智能问答系统等。而图划分是提高搜索性能的关键因素之一，因此如何提高用户的搜索满意度，就是如何快速的寻找图划分的最优解，达到快速响应任务的目的，是面向实际应用的急迫需求。

2 图划分相关理论与技术

2.1 图划分基础理论

图的 k 划分是一个经典的 NP 难问题，在对图划分进行定义之前，我们首先介绍割边。给定两个分区 P_i 和 P_j，其中 $i \neq j$。如果这两个分区中的任意两点存在边，而称这条边为一条割边。符号 s_i 为分区 P_i 的负载 (在有权网络中，负载为分区 P_i 中顶点的权重和。在无权网络中，负载为 P_i 中顶点的数量，没有特殊说明一般都指无权网络)，平衡因子 τ 表示不同分区之间的不均衡程度，$cut-edge$ 表示分区之间总的割边数。$cedge(P_i, P_j)$ 表示分区 P_i 和 P_j 之间的割边数，用数学符号描述为:

$$\begin{aligned} &1-\tau \leqslant s_j/s < 1+\tau, i \leqslant 1,2,\cdots,k \\ &cut-edge = \sum_{j=0,j=0}^{n} \text{cedge}\left(P_i, P_j\right) \end{aligned}$$

图划分，对于给定的图 $G=(V,E)$ 和一个正整数 k，V 代表图中顶点集合，E 代表图中所有边的集合。图划分就是将节点集合 V 划分成互不相交的 k 个集合 $P_1, P_2, ..., P_k$, 每个集合称为一个分区，同时图划分还需要满足两个重要原则。(1) 保证各分区的节点均衡，规模不能相差太大。(2) 割边数尽量的小 (割边就是各个分区之间的边)。对于上述衡量图划分的效果的两个准则，可以定义图划分的目标函数 $obj(P)$ 如下.

$$obj(P) = \begin{cases} \text{minimize}(\tau) \\ \text{minimize}(\text{cut}-\text{edge}) \end{cases}$$

图划分的两种基本划分为点切分和边切分。顶点切分即切分顶点，每条边只保存一次。被切分的顶点会被分发到不同的节点上，增加了存储空间，并且有可能产生同步问题。但是，它的优点是减少了网络通信。边切分即顶点只保存一次，这增加了网络传输的数据量，但好处是节约了存储空间。对于图 2 来说，简单的点切分和边切分如下，左边为点切分，右边为边切分。

2.2 经典图划分算法

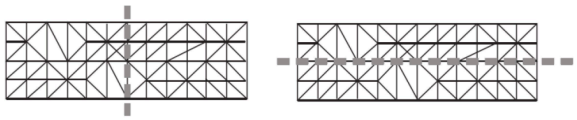
图的 k− 划分广泛应用于图像分割 [32]，数据挖掘 [33]，VLSI 设计 [34] 等领域。研究者对图划分，及其相关的应用进行广泛地研究，从 20 世纪 90 年代至今，不断提出性能较好的图划分算法。目前图划分研究主要分为 3 大类：离线划分，流式划分以及动态重划分。

2.2.1 离线划分

离线划分是将图数据载入内存重复迭代计算，需要频繁访问顶点与边。离线划分主要包括几何划分、谱划分、启发式划分以及多级层次划分。

几何划分法，该方法没有考虑顶点的度，仅根据网格中节点的坐标信息计算出划分方案。虽然几何划分法的划分效率高，但是由于其可利用的信息较少，所以划分质量较差。针对几何划分法的划分质量低问题，研究人员对其进行了改进，提出了一系列的改进算法，包括坐标嵌套二分，空间填充曲线和递归惯性二分。**坐标嵌套划分**，与几何划分法类似，不同的是，坐标嵌套划分法要根据每个顶点的坐标信息计算出它们的质心，同时要将这些质心按照高维 x 轴方向进行投影，最后在 x 轴方向会形成一排有序序列，再对这个有序序列进行二划分，最终输出该网格的二划分结果。图 6 显示了坐标嵌套划分法的一个示意图。两个图分别展示了沿不同轴的二划分，由图可以直接看出图 3 沿 x 轴划分的割边数明显小于图 6 的划分策略，因为该网格的 x 轴方向比 y 轴方向更长。**迭代惯性二分法**，与坐标嵌套划分法类似，所不同的是，迭代惯性二分法要将每个顶点的质心投射到网格的惯性轴上，最后在惯性轴上形成一排有序序列，再对这个有序序列进行二划分，最终输出该网格的二划分结果。在此基础上对子网格进一步迭代二划分，也可以输出该网格的多路划分。由于网格的最长维度不一定是沿着 x 轴和 y 轴。也可能会与坐标轴产生一定的角度。所以这也制约了坐标嵌套划分法的划分质量。**空间填充曲线法**，是一种降低空间维度的近似表示方法，首先将数据空间划分为大小相同的网格，再对这些网格按照一定的方式进行编码且又能保持空间数据的相似性，每个网格的编码不同。如此便将高维空间中的数据降维表示到一维空间，使用线性索引结构存储数据。几何划分法的问题在于没有考虑到网络的拓扑结构以及顶点的度，没有针对具体图进行优化，所以对于图的划分质量不是很理想。

谱划分法，图的划分问题和矩阵的特征向量联



(a)沿垂直方向划分 (b)沿水平方向划分

图 2 两种划分方式

系起来，划分方案通过计算图的离散拉普拉斯矩阵的特征向量来求解。给定一个图 G(V, E) 及图的邻接矩阵 A，其中 A_{ij} 表示顶点 i 和 j 的一条连边。D 为图 G 中顶点度的对角矩阵。d_{ij} 表示顶点 i 的度。拉普拉斯矩阵 L 定义为 $L = D - A$。在以最小化割边数为目标的情况下，拉普拉斯矩阵 L 的第二小特征值 λ_2, 能够生成最佳割边方案的最佳成本的下限 $c \geq \lambda_2/n$。特征向量 V_2 对应了特征值 λ_2，称为 $Fiedler$ 向量，根据对应特征向量，将图划分为两个子图。通过重复二划分或使用与最小特征值相对应的多个特征向量，可以实现图的 k 路划分。虽然谱划分比上述几种方法的划分质量都要好，但是计算拉普拉斯的特征值与特征向量的开销非常大，但是有些学者也对谱方法进行了改进来加速谱划分，通过近似求解特征向量与特征值来降低计算 Fiedler 向量的时间，但是划分的质量却下降了，且划分时间对于大规模复杂网络的划分完全不适用。

启发式分法，启发式算法是解决 NP 难问题常用的方法之一，已有许多研究者利用启发式方法来解决图划分问题。KL 算法是最早的启发式算法之一，通过在两分区中的顶点不断的交换降低割边数，迭代优化最终使割边率降到最低。优化过程其实也是微调方式。具体步骤为：首先将图进行初始划分，可以是随机也可以是哈希方式，接着在满足负载约束的前提下按照增益值又大到小 (必须为正)，不断的交换两分区中的两点，提高划分质量。其中，需要注意的是，每轮次中交换过的顶点不再交换，直到当前轮次迭代停止，开始下一轮次的优化。KL 算法的时间复杂度为 $O(n^2logn)$，对于只有上万顶点的图，也很难处理。随后出现了 KL 算法的改进版本 $FM(Fiduccia-Mattheyse)$ 算法，虽然时间复杂度相对于 KL 有所降低，但是 FM 算法仍无法处理顶点数为百万以上的图，而且 FM 算法很容易陷入局部最优，划分效果很不稳定。由于启发式算法较高的时间复杂度，不适用于大规模图的划分。

多级层次划分法，执行过程主要包括三个步骤，粗糙、初始划分以及细化。粗糙过程主要对原图进行一级一级的压缩，直到压缩为满足条件且规模足够小的图。此图作为粗糙化后最终的小图，对此图进行 k 路划分 (也叫初始划分，可以采用现有的图划分算法，如 KL 算法等)。划分完之后，逐渐将此图还原为原始输入
图，称此过程为细化。在细化过程中，多级层次划分法也采用了一些启发式方法 (例如 Tabu 搜索算法) 进行优化，直到最终得到原输入图的一个划分。图的粗糙过程主要是对原图规模进行一级一级的压缩，直到压缩为满足要求规模足够小的图。每一级粗糙过程中，都会将图中的顶点进行两两合并，一般为随机缩并，将任意相邻的两点进行合并，这个适用于无权网络。但是在有权网络中，效果没有最大边权缩并好，最大边权缩并是将权重最大的邻边两顶点进行合并。通过合并最大的边权 (黑色边)，缩并的效果明显好于随机缩并。由于多级划分算法不管是划分效率还是划分质量都比以上划分算法提高了许多，划分效率高主要是因为多级划分的粗糙过程，将原图中顶点两两合并，直到压缩为满足要求的规模足够小的子图，再对其进行划分。

2.2.2 流式划分

流划分只需要遍历一次就可以完成图的划分，所以流式划分的效率很高。流划分也能够有效处理动态的大图数据，例如 Facebook，Skype 和 Twitter 每天都会有新的账户的创建和删除，用户的每次登入都会通知其它的在线联系人，如果他的大部分好友与他不在同一个服务器上，它们之间将通过不同网络基础设施之间进行通信，通信量明显增多，平衡流算法很好的解决了这一问题，每次只要计算代价函数将新节点放入相应最优值的服务器上，这样通信量就会明显的降低。由于流式划分每次任务的分配可利用的信息很少，所以划分的质量较低。

2.2.3 动态重划分

在分布式图处理系统的运行过程中，不可避免的会出现负载的不均衡或者通信量的增加，例如：在线社交服务器每时每刻都会有新用户的注册与老用户的注销，集群某些硬件的升级等会导致集群负载的不均衡，某个计算节点的故障等会导致通信量增加。很多研究者们针对这些问题展开了一系列的研究，提出了一些性能较好动态图划分算法，一种非结构化网格进行动态划分的并行方法。该方法引入了一种新的迭代优化技术，称为相对增益优化，既平衡了工作负载，又尽量减少了处理器间的通信开销。对一系列自适应优化网格的实验表明，

该算法为静态分区提供了同等或更高质量的分区，而且速度更快。也许更重要的是，与静态分区相比，该算法只导致数据迁移量的一小部分。在论文中提出了一种自适应的方法，在这种方法中，图形会随着每次更改而优化，而不是计算执行。通过基于分散迭代顶点迁移的局部决策启发式方法，以可扩展的方式改进了图分区。该方法在试图最小化切割边数的分区之间迁移顶点，同时在运行时根据结构更改保持分区平衡。

3　研究进展

3.1　Gemini 的块式划分

图数据的划分是所有分布式图计算系统的核心。然而，已有的划分方法主要关注负载均衡和通信代价，往往忽视了因此产生的系统复杂度和对计算效率的影响。Gemini[23] 采用了一种十分简单的划分方法: 将顶点集进行块式划分，将这些块分配给各个节点，然后让每个顶点的拥有者 (即相应节点) 维护相应的出边 / 入边。图 3 展示了有向图划分给两个节点的结果，10 个顶点中的 0～4 分配给了左边的节点，5~9 分配给右边的节点。图中的白色顶点表示拥有的顶点 (即主备份)，黑色顶点则为镜像备份。这种划分方式看似简单，但是应用在很多现实世界的图上却格外有效。这是由于很多图数据本身蕴含了局部性，而块式划分很好地保留了这些特征。

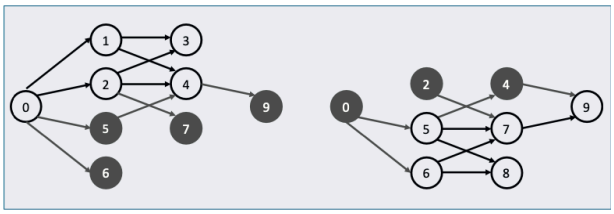


图 3　块式划分

其次，块式划分的另一大优点是极小的分布式开销。其他划分方法依据策略的不同极有可能将连续的顶点划分到不同节点上，通常需要在各个节点上维护映射表，将每个节点负责的顶点编号从较为分散的大区间映射到紧凑的小区间内。块式划分则由于所有节点负责的均是连续的一块顶点，从而免去了顶点编号转换的开销。

块式划分由于块内顶点的连续性，还可以自然地 “递归” 应用到更细粒度上。例如，现有的多处理器通常呈现非一致内存访问效应 (Non-Uniform Memory Access)，尽管共享所有的内存，但是处理器访问不同区域的内存会有不同的延迟和带宽，对于各个处理器而言，有本地内存和远程内存的区别。因此，Gemini 在每个节点的多路处理器间继续进行块式划分，可以有效减少远程内存的访问比例。

传统的图划分方法根据划分对象为顶点或边进行负载均衡，让每个节点分配到的顶点数或边数尽可能相近。然而，我们发现这两种策略应用在块式划分上均有不同的缺陷。按顶点数均衡可能导致边数失衡，而让边数均衡时顶点数又会有潜在的较大差异。两者都会影响计算效率: 边数决定了计算量，而顶点数决定了单次计算 (即随机内存访问) 的复杂度。因此，Gemini 采用了一种混合的策略，让每个节点分配的顶点数和边数的加权和尽可能相近。

3.2　Gluon 的图划分

Gluon[24] 实现的分区策略可以统一呈现如下。图的边使用某种策略分布在主机之间。如果为主机分配了一条边 (N1,N2)，则会为节点 N1 和 N2 创建代理，并通过该主机上的一条边进行连接。如果连接到给定节点的边最终位于多个主机上，则该节点在分区图中具有多个代理。一个代理被指定为该节点所有代理的主节点，其他代理被指定为镜像。主节点负责在同步期间保存和传达节点的规范值。

在某些分区策略对于某些操作符可能不合法的意义上，分区策略可以与操作符的结构交互。为了理解这种交互，将图划分的各种策略（或启发式）分为四个类别或策略是很有用的。这些将在下面描述并在图 4 中说明。在该图中，图中的节点 N 在不同的主机上具有三个代理；阴影代理是主。

- 无约束顶点切割 (UVC) 可以将节点的传出和传入边分配给不同的主机。
- 笛卡尔顶点切割 (CVC) 是一种受约束的顶点切割：只有 master 可以同时具有传入边和传出边，而镜像代理可以具有传入边或传出边，但不能同时具有两者。
- 入边切割（IEC）受到更多限制：传入边缘仅分配给主节点，而节点的传出边缘在主机之间进行分区。
- 出边切割 (OEC) 是 IEC 的镜像：传出边缘仅分配给主机，而传入边缘在主机之间进行分区。

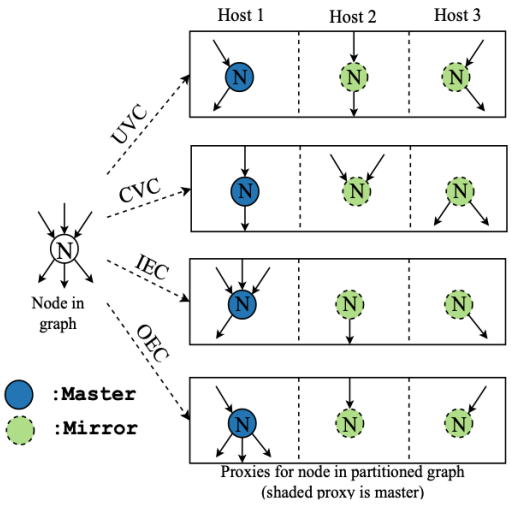


图 4 Gluon 的四种划分方式

只有当运算符具有特殊结构时，才能使用其中一些分区策略。对于拉式算子，只有当算子对活动节点标签所做的更新是减少时，才能使用 UVC、CVC 或 OEC；否则，必须使用 IEC，因为本地计算所需的所有传入边都存在于主设备上。对于推送式算子，UVC、CVC 或 IEC 仅在节点沿其输出边推送相同值并使用归约将推送到其传入边上的值组合在一起时才能使用；否则，必须使用 OEC，因为主节点具有计算所需的所有传出边。本文研究的基准满足所有这些条件，因此可以对它们使用任何分区策略。

3.3 XtraPuLP 大规模图划分器

基于可扩展的标签传播社区检测技术，该技术已在各种先前的工作中被证明是一种可行的方法，可以以最少的计算时间生成倾斜和小世界图的高质量分区。在大型稀疏图的集合上，XtraPuLP[25] 分区比最先进的分区方法快得多，可以生成具有十亿多个顶点和超过一千亿条边的真实世界图的分区。XtraPuLP 主要分为四个阶段，顶点平衡，顶点优化，边平衡，边优化。

顶点平衡阶段，在图增长初始化阶段允许相当大的不平衡，以确保至少有一部分到达大多数顶点；否则零件无法到达的顶点被随机分配给一个。然而，即使有很大的不平衡，这个后续的平衡阶段也可以有效地重新平衡初始部分。**顶点优化阶段**，此 XTRAPULP 细化阶段贪婪地最小化切割边缘的全局数量，而不会超过顶点目标零件尺寸如果在平衡阶段已满足约束）或不增加任何大于当前最不平衡零件的零件尺寸。这种细化算法可以被认为是 FM 细化的变体或基线标签传播的受限变体。在顶点平衡细化阶段的外部迭代次数之后，**边缘平衡细化阶段**开始。通过考虑每个等级的目标边数和顶点数来进行平衡。目标是平衡每个等级的边数，同时不造成顶点不平衡。**边优化阶段**，与顶点优化阶段类似，贪婪地最小化切割顶点的全局数量。XtraPuLP 通过最小化最大每部分切割来平衡切割的能力使其对许多在通信期间利用全重量级全对全交换的图应用程序特别有用。

3.4 平衡标签传播 BLP 算法

平衡标签传播 (BLP[26]) 对围绕半监督学习和社区检测的标签传播文献进行了约束。BLP 算法对节点集的初始可行分区（标记）进行迭代、平衡的改进，直到达到平衡（或达到最大迭代次数）。在这项工作中，我们使用最简单的初始化——随机平衡分配——与其他方法进行比较，尽管仔细的初始化已被证明可以实现更好的平衡切割，但这取决于上下文和可用的元数据。

每次迭代进行如下：对于每个节点 u □ V，计算它的移动增益，如果单边重新定位，则共同定位的邻居计数的最大改进，定义为

$$g_u = \max_{i\in[k]} \left|N(u)\cap V_i\right| - \left|N(u)\cap V_{P(u)}\right|$$

对于所有的 $u \in V$，显然是 $g_u \geq 0$。当 $g_u = 0$ 时，节点 u 有效地“满足”，并保持其分片赋值，这是我们在第 4 节中形式化的概念。当 g_u□0 时，将节点放入队列中，按增益递减的顺序移动到目标分片。这些信息被导入一个线性程序中，该程序在迭代过程中解决循环问题，确定从这些队列中移动的最大顶级节点数，从而在遵守每个分片大小的约束的同时获得最大的收益。为所有分片对执行这些节点重新定位构成一个迭代，并且该过程重复，直到没有节点想要移动，或者达到最大迭代次数为止。

3.5 Social Hash Partitioner

SocialHashPartitioner□*SHP*□[27] 是一个用于生成和更新图数据分区的两级框架，是为优化脸谱网的 *SocialHash* 基础设施而开发的。它被用来划分更一般的超图数据，最小化一个叫做 *fanout* 的目标（一个超边跨越的平均分片数）。该算法很容易“扩展”到传统的边割目标下的非超图划分，尽管文献中尚未对此进行评估。其均衡分片大小的机制是最早的均衡分割算法之一 *Kernighan–Lin* 算法的自然 *k* 向扩展，该算法用于最小化边缘切割。

与 BLP 一样，SHP 从节点集的初始划分开始，对边切割目标进行迭代改进，直到达到平衡或最大迭代次数。论文中提出的 SHP 的原始实现方式如下：计算每个节点 $u \in V$ 的增益，如下所定义。当增益大于 0 时，节点按指数大小存储在容器中，移动到目标分片 t_u，每个分片对存储两个直方图。对于所有对，这些存储桶被确定地配对并从最高增益交换到最低增益，直到最后一个存储桶，其中节点以随机顺序交换，直到没有更多的交换可以增加重新定位的总体增益。

作为一个简化步骤，在这项工作中，我们修改了算法，以存储每个分片对的两个完全排序的节点队列。然后，按照最大增益的顺序对单个节点进行配对和交换，与其中更容易分布的实现相比，这是迭代内目标的严格改进。由于这项工作研究的是这些设计决策对目标的影响，而不是分布式实现的计算权衡，因此这种简化使我们能够以 “最佳” 形式研究 SHP 算法。同时，我们承认迭代中更好的性能并不一定转化为平衡中更好的性能。

作为另一个重要的修改，我们在本例中定义了增益，以将满足要求的节点（其中，u0 的节点）包括在移动队列中，以获得它们的第二个最佳分片，并通过修改的增益形式有效地对外部分片进行排序：

$$g'_u = \max_{i \in [k] \setminus P(u)} \left|N(u) \cap V_i\right| - \left|N(u) \cap V_{P(u)}\right|$$

为了以后的参考，论文将其最佳外部分片的移动队列中的满意节点表示为 “次优” 节点。这些节点包括在自己的费用，可能允许交换与净正的全局收益，一个标志性的特点，最初的 1970 年 Kernighan-Lin 算法。因此，需要将用 “KL-SHP” 来表示这一明确的实施。

在该文的工作中，$KL-SHP$ 的两个简化表示为 “$SHP-I$” 和 “$SHP-II$”，也用于研究成分设计决策的影响。在 $SHP-I$ 中，我们都将次优节点从重新定位队列中排除，并放弃优先排序，随机配对要交换的节点，直到一个队列为空。在 $SHP-II$ 中，我们排除了次优节点，但仍按排序顺序交换，将交换限制为仅涉及具有正移动增益的节点。比较 $KL-SHP$ 和 $SHP-II$ 显示了局部负（KL 风格）互换的效果；在 $SHP-II$ 和 $SHP-I$ 之间，有序排序的效果。

3.6 重组线性确定性贪婪算法

重组线性确定性贪婪算法 $reLDG$[28] 属于迭代算法的一个称为（再）流算法的子类。这个类是由单通道在线图加载的上下文驱动的，在这个上下文中，程序解析一个图文件，串行地将图数据从源读取到目标集群。论文中提出了该方法的多通道/迭代版本，考虑了用于分区的重新分组方法可能可以与离线、非流式算法竞争。

$reLDG$ 算法源自 LDG，将节点列表重复进行流传输，直到达到最大迭代次数。具体来说，$reLDG$ 在每次迭代时都执行以下操作：对于每个 $u \in V$，将 u 赋给满足以下条件的分片

$$\arg\max_{i \in [k]} \left|V_i^{(t)} \cap N(u)\right| \cdot \left(1 - \frac{x_i^{(t)}}{C}\right)$$

这里 $V_i^{(t)}$ 保存分片 i 的当前规模，来自上一个或当前流（如果适用的话，如果节点已经 “看到” 了这个迭代），$x_i^{(t)}$ 保存当前流中分配给 i 的节点数量，C 是分片容量约束，$C = (1+\epsilon) \cdot \lceil \frac{|V|}{k} \rceil$。注意，当分片开始填充时，“乘数权重”$1 - x_i^{(t)}/C$ 接近于零，最终将填充的分片从考虑中删除。

毋庸置疑，流顺序中节点的位置对该节点周围生成的分区的质量起着很大的作用。流开始处的节点尚未受到乘法权重的影响，而流结束处的节点分配可能受该项支配。先前对 LDG 和 $reLDG$ 的研究主要集中在随机（持久）顺序上，在最初的 LDG 工作中考虑了 BFS/DFS 顺序。

3.7 应用驱动的图划分

先考虑两个经典的算法模型。**公共邻居算法**。考虑在图 5(a) 中的有向图运行公共邻居。公共邻居计算的是每一对顶点的共同邻居的数量。广泛应用于链路预测，产品推荐和碰撞检测。为了简化讨论，我们只考虑出边共同邻居，即对于一个顶点 u 来说，u 是 v_1 和 v_2 的共同邻居，那么即存在 v_1 到 u 的出边和存在 v_2 到 u 的出边。所以公共邻居算法如下工作，即对于每个三元组 (u, v_1, v_2) 来说，v_1 和 v_2 都有指向 u 的边。假设图 G_1 被划分为图 5(b) 中的分区 F_1 和分区 F_2，那么分区在顶点和边数上是完全平衡的，每个分区都有 5 个实点和 9 个边。

图 5(b) 似乎是 balanced，因为每个 Fragment 都有 **5 个顶点，9 条边**。但是对于 CN 程序来说，每个顶点 u $(u, v1, v2)$：　$v_1 \longrightarrow u$　$v_2 \longrightarrow u$，至多贡献 $\frac{1}{2}d^+(u)(d^+(u)-1)$ 个三元组，$d^+(u)$ 为顶点 u 的入度。所以实际上 CN 的计算负载为 $\sum_{u \in F_i} \frac{1}{2}d^+(u)(d^+(u)-$

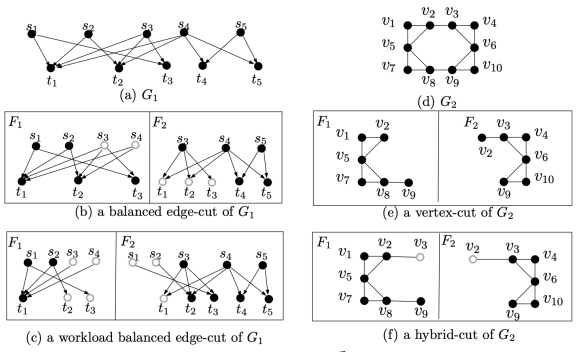


图 5 图分区例子

1) . 而对于图 5(c) 是 G_1 的另一个分区。虽然看起来没有图 5(b) 中的分区更平衡，因为分区 F_1 有 3 个顶点，6 条边，分区 F_2 有 7 个顶点，11 条边，但是，在两个分区上公共邻居的工作负载都是 6. 这表明顶点和边平衡等静态指标并不能确保 CN 等应用程序的工作负载平衡。

三角形计数。同样，虑在图 5(d) 中的无向图运行数三角形。三角形计数的程序按如下：对于每个节点 v, 以及其边 (u, v), 程序根据边 (v, u) 计算三角形的数量。图 5(e) 按节点划分，分区之间需要通信，假设划分 $v_2, v_9, N = \{v_1, v_3, v_5\}$ 是 v_2 的邻居，计算包含 v_2 的三角形，必须检查 $N \times N$ 中所有的顶点对，需要跨节点通信检查 (v_2, v_3). v_9 类似。副本可以有效减少通信成本。考虑图 5(f) 中的图，$(v2, v3)$ 可以在 F_1 中存在副本 v_3，同理 F_2。当三角形计数运行在 v_2 上时，所有的操作都可以在本地做，不需要额外的通信成本。

除此之外，在边缘切割分区下开发的算法可能无法在顶点切割下工作，反之亦然。因此，在为图计算问题开发并行算法时，必须提前选择要使用的分区策略。如果选择的策略效果不佳，我们可能不得不重写我们的算法并切换到另一个。基于以上的几个角度，提出了应用驱动分区策略——专为算法 $\mathcal{A}$ 的**混合分区策略**，引入了算法的计算和通信模式的**代价模型**，公式化了 ***ADP***（application-driven partition problem），旨在寻找代价减少的分区。ADP 是完全 np 难问题。代价函数学习采用**多项式回归**，证明了实用性。**SGD 算法**训练模型。学习到的成本模型可以应用于 A 运行的不同图。给定算法 A 和图 G，在 A 的学习成本模型的指导下为 A 开发 G 的混合分区。ParE2H(ParV2H) 提炼了 edge-cut(vertex-cut) 到一个混合分区，以适应 A 的成本模式。

论文[29] 提出了一个应用程序驱动的分区策略。对于给定的图算法 A，展示了如何学习它的成本模型，并开发了分区器来细化边切割或顶点切割分区以适应 A 的成本模式并加速 A 的并行执行。

4 总结与展望

4.1 总结

虽然已提出了许多图划分方法，但这些方法并不是尽善尽美，仍然存在某些问题。比如，现有的图划分算法大部分是解决不带权图（顶点 / 边的权值相等）的划分问题。然而在某些实际的应用场景中，每个顶点 / 边的权值可能不相等。例如，在社会网络中，某些边的关系较强，表示它们有着十分频繁的用户交互，如果简单地认为关系强的边和关系不强的边是相等的，那么便脱离了社会网络的实际用户交互情况。可见，带权图的划分模型在某些实际的应用场景中更实用。尽管前面已经有针对具体算法的具体应用程序驱动的分区策略。对于给定的图算法 A，我们已经展示了如何学习它的成本模型，并开发了分区器来细化边切割或顶点切割分区以适应 A 的成本模式并加速 A 的并行执行。但是针对大规模图的算法而言，具体应用驱动的分区策略从头开始的预处理，会很慢，而且由于分区的点和边可能极度不均衡，其个节点的内存开销和访存开销，各节点之间的通信同步等都会存在很大的问题。

4.2 展望

正如计算科学的发展来看，计算资源计算科学的发展，必然是分布式与单机并行，目前大数据时代，集成电路已经到达瓶颈，新的计算架构，新的运算芯片还未诞生，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，分布式并行环境下大规模图数据的划分算法的研究日益迫切。通过图形数据处理进行大数据分析是业界和研究界非常感兴趣的问题。图形数据提供了集成和处理任何类型复杂数据的灵活性。大规模图划分是一个 NP-Hard 问题，它对时间复杂度、工作负载、负载均衡和网络带宽都提出了挑战。大规模图的划分问题今后仍然并长期是一个热点，主要围绕的关键点现在已经不是图划分的计算代价，因为分布式系统云计算等平台的诞生已经可能很好的得到利用资源，图计算的加速瓶颈在今后很大程度上是来自于分布式系统的通信问题。

致 谢 此文在施老师的亲切关怀和悉心指导下完成的。他严肃的科学态度，严谨的治学精神，精益求精的工作作风，深深地感染和激励着我。从课题的选择到项目的最终完成，老师都始终给予我细心的指导和不懈的支持。在此谨向老师致以诚挚的谢意和崇高的敬意。

Yujian Liao,

参 考 文 献

[1] Berger M J, Bokhari S H. A Partitioning Strategy for Nonuniform Problems on Multiprocessors [M]. 1987

[2] Ou C W, Ranka S. Parallel incremental graph partitioning using linear programming [M]. 1994.

[3] Xu N, Cui B, Chen L, et al. Heterogeneous Environment Aware Streaming Graph Partitioning [J]. IEEE Transactions on Knowledge and Data Engineering, 2015, 27(6): 1560-72.

[4] Ou C W, Ranka S, Fox G. Fast and parallel mapping algorithms for irregular problems [J]. Journal of Supercomputing, 1996, 10(2): 119-40.

[5] Warren M S, Salmon J K. A parallel hashed oct-tree N-body algorithm [C]. proceedings of the conference on high performance computing (supercomputing), 1993, 12-21.

[6] Pothen A, Simon H D, Liou K P. Partitioning sparse matrices with eigenvectors of graphs. SIAM J Matrix Anal Appl [J]. Siam Jmatrix Analappl, 1990, 11(3): 430-52.

[7] Barnard S T, Simon H D. Fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems [J]. Concurrency & Computation Practice & Experience, 2010, 6(2): 101-17.

[8] George A, Liu J W H. Computer Solution of Large Sparse Positive Dejnite Systems [M]. 1981.

[9] Pothen A, Simon H D, Wang L, et al. Towards a fast implementation of spectral nested dissection [C]. proceedings of the Supercomputing, 1992.

[10] Hendrickson B, Leland R. An Improved Spectral Graph Partitioning Algorithm for Mapping Parallel Computations [M]. 1992.

[11] Fiduccia C M, Mattheyses R M. A Linear-Time Heuristic for Improving Network Partitions [C]. proceedings of the Papers on Twenty-five Years of Electronic Design Automation, 1988.

[12] Mykhailenko H, Neglia G, Huet F. Simulated annealing for edge partitioning [C]. proceedings of the conference on computer communications workshops, 2017, 54-59.

[13] Leng M, Yu S. An effective multi-level algorithm based on ant colony optimization for bisecting graph [C]. proceedings of the knowledge discovery and data mining, 2007, 138-149.

[14] Hager W W, Park S C, Davis T A. Block Exchange in Graph Partitioning [M]. 2000.

[15] Cong J, Smith M. A Parallel Bottom-up Clustering Algorithm with Applications to Circuit Partitioning in VLSI Design [C]. proceedings of the design automation conference, 1993, 755-760.

[16] Karypis G, Kumar V. Parallel Multilevel k-way Partitioning Scheme for Irregular Graphs [C]. proceedings of the conference on high performance computing (supercomputing), 1996, 35.

[17] Heuer T, Sanders P, Schlag S. Network Flow-Based Refinement for Multilevel Hypergraph Partitioning [J]. 2018.

[18] Stanton I, Kliot G. Streaming graph partitioning for large distributed graphs [C]. proceedings of the Acm Sigkdd International Conference on Knowledge Discovery & Data Mining, 2013, 1220-1230.

[19] Zheng W, O'Boyle M F P. Partitioning streaming parallelism for multi-cores:a machine learning based approach [C]. proceedings of the International Conference on Parallel Architectures & Compilation Techniques, 2010, 307-318.

[20] Abbas Z, Kalavri V, Carbone P, et al. Streaming graph partitioning: an experimental study [J].
very large data bases, 2018, 11(11): 1590-603.

[21] Kumar D, Raj A, Dharanipragada J. GraphSteal: Dynamic Re-Partitioning for Efficient Graph Processing in Heterogeneous Clusters [C]. proceedings of the international conference on cloud computing, 2017.

[22] Khayyat Z, Awara K, Alonazi A, et al. Mizan: a system for dynamic load balancing in large-scale graph processing [C]. proceedings of the european conference on computer systems, 2013, 169-182.

[23] Xiaowei Zhu, Wenguang Chen, Weimin Zheng, and Xiaosong Ma. 2016. Gemini: A Computation-centric Distributed Graph Processing System. In Proceedings of the 12th USENIX Conference on Operat-

ing Systems Design and Implementation (OSDI'16). USENIX Association, Berkeley, CA, USA, 301–316

[24] Roshan Dathathri, Gurbinder Gill, Loc Hoang, Hoang-Vu Dang, Alex Brooks, Nikoli Dryden, Marc Snir, and Keshav Pingali. 2018. Gluon: A Communication-Optimizing Substrate for Distributed Heterogeneous Graph Analytics. In Proceedings of 39th ACM SIG- PLAN Conference on Programming Language Design and Imple- mentation (PLDI'18). ACM, New York, NY, USA, 17 pages.

[25] Slota G M, Root C, Devine K, et al. Scalable, multi-constraint, complex-objective graph partitioning[J]. IEEE Transactions on Parallel and Distributed Systems, 2020, 31(12): 2789-2801.

[26] Ugander and L. Backstrom. 2013. Balanced Label Propagation for Partitioning Massive Graphs. In WSDM. 507–516.

[27] I. Kabiljo, B. Karrer, M. Pundir, S. Pupyrev, and A. Shalita. 2017. Social hash partitioner: a scalable distributed hypergraph partitioner. VLDB 10, 11 (2017).

[28] J. Nishimura and J. Ugander. 2013. Restreaming Graph Partitioning: Simple Versatile Algorithms for Advanced Balancing. In KDD. 1106–1114.

[29] Wenfei Fan, Ruochun Jin, Muyang Liu, Ping Lu, Xiaojian Luo, Ruiqi Xu, Qiang Yin, Wenyuan Yu and Jingren Zhou. 2020. Application Driven Graph Partitioning . In Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data (SIGMOD'20), June 14–19, 2020, Portland, OR, USA. ACM, New York, NY, USA, 15 pages.