大规模图划分方法研究综述

熊益[1)]

[1)](华中科技大学计算机科学与技术学院，武汉 430074)

摘 要 图是计算机学科中很常用的一种数据结构，其顶点可以表示一个具体的对象，边可以用来表示对象之间的相互关系，因此图被广泛用于理解和分析许多不同领域中关系。随着网络的快速发展，以网络为载体的信息量迅速增长，使得图的规模也随之扩大，由于大规模图数据无法使用单机环境进行处理，因此就需利用分布式图计算将图数据合理地分布到每个节点，而图划分问题的研究也随着实际应用的需求不断驱动。现有的图划分方法是为轮询同步分布式框架而设计的，它们平衡了工作负载，而不区分顶点的重要性，也没有考虑到基于优先级的调度的特征；许多图形应用程序将其数据存储在地理分布式数据中心上，以在世界范围内提供低延迟的服务，这也就意味着分布式数据中心在网络带宽和通信价格方面存在多层异构；传统的图划分算法在处理大规模幂律图时，由于巨大的通信量或分区子图之间极端的消息不平衡，在许多以子图为中心的框架中都存在通信瓶颈。针对以上三个挑战，本文通过梳理相关研究工作，分别介绍其解决方法：热平衡划分（HBP）、Geo-Cut划分、针对幂律图的高效平衡顶点划分算法（EBV），并深入分析每个方法的原理和算法细节。

关键词 图划分；复杂网络；并行图计算 ；负载均衡；

Survey on Large-Scale Graph Partitioning Methods

Yi Xiong[1)]

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

Abstract Graph is a kind of data structure commonly used in computer science. Its vertices can represent a specific object and edges can be used to represent the relationship between objects. Therefore, graph is widely used to understand and analyze the relationship in many different fields. With the rapid development of network, network as the carrier of information rapid growth, the figure of scale expansion, as due to large-scale figure data cannot be used standalone environment, so we have to distributed diagram is used to calculate the figure data reasonably distributed to each node, and the graph partition problem research is as the demand of practical application continues to drive. Existing graph partitioning methods are designed for polling synchronous distributed frameworks, which balance workloads without differentiating the importance of vertices or taking into account the characteristics of priority-based scheduling; Many graphics applications store their data in geographically distributed data centers to provide low-latency services around the world, which means that distributed data centers have multiple layers of heterogeneity in terms of network bandwidth and communication prices. When traditional graph partitioning algorithms deal with large power law graphs, communication bottlenecks exist in many subgraph-centered frameworks due to huge traffic or extreme message imbalance between partitioned subgraphs.In view of the above three challenges, this paper proposes different solutions: heat balance partition (HBP), Geo-cut partition and efficient balanced vertex partition algorithm for power law graph (EBV) through sorting out relevant research work, and analyzes the principle and algorithm details of each method in depth.

Key words Graph Partitioning; Complex Networks; Parallel Graph Computing; Load Balancing

1 引言

为了处理海量的图，分布式图处理系统将图数据划分为多个图分区，并在一组节点上进行处理，每个节点处理一个图分区。但在分布式图计算过程中，由于存在大量的边/顶点切割，节点之间的通信开销较大；由于工作负载不均衡，节点之间可能存在空闲，从而降低了分布式计算的性能。为了降低通信成本和空闲节点，很多研究在寻找智能图划分方法，作为大规模分布式图处理的预处理步骤，旨在最小化图分区之间的连接，使节点负载均匀地分布在分区之间。

之前的图划分算法都假设底层的分布式图挖掘框架采用同步并行处理模型。在同步并行模型中，每次迭代后都存在一个全局同步障碍。在每个节点上，分配的顶点以循环方式处理。在循环调度中，顶点按循环顺序处理。所有的顶点计算在每一轮被调用，没有区别。由于循环操作简单、易于实现、无饥饿，因此得到了广泛的应用。在每次迭代中，每个顶点只处理一次，每条边只传递一条消息。因此，图分区的顶点(对于以顶点为中心的框架)或边(对于以边缘为中心的框架)的数量表示节点的工作负载，而切割边(或顶点切割系统中的顶点副本)的数量表示工作者之间的通信成本。但如果节点的工作量不均衡，轻负荷的节点会等待重负荷的节点。此外，大量的切割边(或顶点复制)将导致显著的网络开销。近年来，有许多研究工作关注异步并行处理。在这些异步分布式框架中，消除了全局同步障碍。因此节点之间没有等待时间，顶点/边可以在任何时候被处理。换句话说，没有这样的约束，即每个顶点/边在每次迭代中只能被处理一次，并且某些顶点/边可以被处理更多次。一些顶点确实在决定最终收敛结果中发挥了重要的决定性作用。在异步计算中，顶点的重要性是不一致的。在优先级处理中，每一轮只选择一个顶点子集(即重要顶点)进行处理，因此这些重要顶点的处理频率高于其他顶点。

同时许多图形应用程序(如社交网络)都涉及分布在多个地理分布(地理分布)数据中心中的大型数据集。例如，Facebook 每天都会从世界各地的用户那里接收到海量的文本、图像和视频数据。为了向用户提供可靠、低延迟的服务，Facebook 建立了四个地理分布数据中心来维护和管理这些数据。有时，由于隐私和政府监管的原因，无法将数据移出数据中心。因此，对这些数据进行地理分布处理是不可避免的。这也就意味着跨地理分布式数据中心划分和处理图形数据的许多技术挑战。首先，在不同的图处理算法下，图处理过程中的流量模式可以是高度异构的。一方面，不同的顶点传输的数据大小会有很大的差异，这主要是因为顶点的程度不同。在自然图中，顶点的度通常服从幂律分布，这导致在处理图时，一小部分顶点消耗了大部分流量。另一方面，根据图处理算法的不同特点，单个顶点的发送和接收数据大小也可以是异构的。

并且尽管顶点中心模型已经成功地应用于许多并行图计算应用程序中，但由于通过网络交换过多的消息，它在通信方面存在显著的瓶颈，以子图为中心的模型为并行图计算提供了另一种方法，并已在一些最先进的平台中得到应用。与以顶点为中心的模型相比，以子图为中心的模型关注的是整个子图。这个模型被称为“像图表一样思考”。由于子图比单个顶点更“粗粒度”，它们保留了许多内边，因此在通过网络传输时省略了许多消息。使得以子图为中心的模型通常通信更少，收敛速度更快。但现有的图划分算法对顶点度分布是长尾的幂律图进行划分存在困难，以子图为中心的模型处理大规模的真实世界的图时，并没有充分发挥其潜力，因此需要提高以子图为中心的框架在大规模幂律图上的性能。

2 原理与优势

2.1 热平衡划分(HBP)

传统的图划分工作大多都假定了轮询同步图处理(简称轮询处理)。之前的工作对于优先级异步处理(缩写为优先级处理)来说远非理想。在设计有优先级处理的图划分方法时，需要注意有优先级处理和循环处理的两个关键区别。

首先，工作负载平衡并不一定意味着高效率。随着同步障碍的消除，不同机器之间没有等待时间，也没有空闲的 workers。所有的 workers 都一直在忙着加工。然而，一些顶点在帮助迭代计算的收敛方面发挥了重要作用。这些重要的顶点需要比其他顶点进行更多的更新。为了提高计算效率，图的划分方法应该考虑到这一特性。

其次，少量的边缘切割或顶点复制并不一定意味着更少的通信成本。由于优先级调度，每个顶点

上的更新数量不一致，通过每个边缘的消息数量也不一致。即使有很少的边缘切割/顶点复制，它仍然可能导致大量的通信成本，因为沿每条边传递的消息数量是不相同的。如果高优先级顶点的一条边被切断，仍然会有大量的消息沿着这条边传递。

综上所述，优先图计算导致了顶点热度的区别。热顶点的状态更新得更频繁，从这些热顶点传播的消息也更多。现有的针对循环处理的图分区方法不适用于优先级处理。因此，需要设计一种新的分区方法来处理优先级图的贡献。本文提出了热度平衡划分(HBP)的思想，根据顶点的热度对图进行划分。

首先估计每个顶点的热度，然后提出了三个目标的图划分优先处理。为了有效地解决多目标优化问题，提出了一种启发式的基于流的图划分算法SPb-HBP，该算法只需要一次图数据。进一步提出了 SPb-HBP 的分布式版本，即 DSPb-HBP，用于对海量图数据进行分区。

通过比较哈希分区和两种最先进的图分区方案 Fennel 和 HotGraph 进行了实验。结果表明，SPb-HBP 算法能够有效地处理优先级迭代图。特别是，SPb-HBP 通过哈希分区可以减少 40-50%的运行时间，通过 Fennel 可以减少 5-75%的运行时间，通过 HotGraph 可以减少 22-31%的运行时间。

2.2 Geo-Cut划分

文章提出了一种地理感知的图划分方法Geo-Cut。Geo-Cut 考虑了图流量和地理分布式数据中心的特点，旨在在满足数据中心间数据通信成本预算约束的前提下，通过最小化数据中心间数据传输时间来优化图处理性能。

为了提高优化效率，Geo-Cut 采用了两阶段优化方法。

在第一阶段，提出了一种以最小化跨数据中心数据传输代价为目标的流启发式算法，并利用一遍流划分方法快速地将边分配到不同的数据中心。

在第二阶段，提出了两个分区优化启发式算法，识别图处理的性能瓶颈，并对第一阶段得到的分区结果进行优化，在保证预算约束的情况下减少数据中心间的数据传输时间。

由于了两阶段的优化，Geo-Cut 展示了一个轻量级的运行时开销，并且可以轻松有效地扩展到分区动态图。我们用五种现实世界图和三种不同的图算法来评估 Geo-Cut 算法的有效性和效率，并将其与四种最新的图划分方法进行比较。评估结果表明，与其他方法相比，Geo-Cut 可以减少 79%的 DC 间数据传输时间(中位数为 42%)，并减少 75%的货币成本(中位数为 26%)。关于开销，Geo-Cut 只需不到 3 秒划分 100 万条小尺寸的图形，如 GoogleWeb(约10^6边)和大约 100 秒的大尺寸图，如 Twitter (约10^9边)

2.3 EBV划分

现有的图划分算法对幂律图进行划分存在困难，由于幂律图的度分布是倾斜的，因此每个顶点的关联边数变化很大。因此，对于幂律图，不能通过均匀分配顶点和边来实现顶点和边的平衡。

为了提高子图中心框架在大规模幂律图上的性能，文章分析了几种不同划分算法的通信模式。在此基础上，提出了一种高效且平衡的顶点切割图划分算法(EBV)。

EBV 根据设计良好的评价函数的当前值对每条边进行赋值，该评价函数既考虑了切割顶点的总数，又考虑了图划分结果的均衡性。为了处理幂律图的倾斜度分布，在划分时采用适当的权值来平衡边数和顶点数。

此外，文章设计了一种边排序预处理过程，在分区前，按照边的端顶点度数的总和对边进行升序排序。

对于幂律图，首先分配具有两个低阶顶点的边。因为它们的端点顶点的度数很低，所以它们不太可能共享同一个端点顶点。因此，图划分结果的平衡是早期的主要因素，这些低度的顶点作为种子均匀地分配给每个子图。从理论和实验两方面分析了评价函数和边缘排序预处理过程的效果。的实验表明，EBV 在大规模幂律图上的分割算法优于目前最先进的分割算法。

总的来说文章做出了以下贡献：

（1）提出了一种高效、平衡的顶点切分图划分算法。

（2）详细比较了 EBV 与最先进的图划分算法和几种并行图计算框架。

（3）使用通信消息的数量作为独立于平台的度量来比较分区算法。

（4）研究了排序预处理对 EBV 算法的影响。

3 研究进展

3.1 热平衡划分(HBP)

3.1.1 热平衡划分(HBP)思想

传统的分区方法不能满足优先级调度框架的要求。在优先级调度框架中，一些顶点具有较高的执行优先级，因此成为优先级调度系统中的热点顶点。顶点更新的频率称为顶点热度。

给定热度的图$G=(V,E,H)$,以及分区k,其中每个顶点与热值h_i相关联，$H=\{h_v,v\in V\}$为所有顶点的热值，边切图划分的目的是寻找一种分区方案$\mathcal{G}=\{G_1,G_2,\ldots,G_k\}$，其中$G_i=(V_i,E_i,H_i)$为$G$的一个分区，$V_i$是$G_i$的顶点集，使得$V=\bigcup_{i=1}^{k}V_i$，并且$V_i\cap V_j=\emptyset$，$E_i$是源顶点$V$中边的集合，接下来，每个分区被分配给一个 worker 进行并行处理。

热值平衡划分的前提是求得顶点的热值和顶点之间的通信开销。但这些在计算开始之前是未知的，因此，首先介绍了如何估计顶点的热度和顶点之间的通信成本。

顶点的优先级由其Δ_v值决定，Δ_v是从顶点的邻居IN_v中收集的，因此如果顶点v有很强的收集Δ_v的能力，该顶点很可能具有跟高的执行优先级，可能是很热的，因此用下面的公式估计顶点的热度。

$$h_v=\sum_{u\in IN(v)}\frac{w_{u,v}}{\sum_{w\in OUT(u)}w_{u,w}}$$

其中$w_{u,v}$为边的权值(u,v),，如果没有权值，则假定$w_{u,v}=1$。

在计算过程中，当一个顶点被更新时，它将向它的输出邻居发送消息，因此通过一条边的消息数量与源顶点的更新时间成正比。换句话说，边缘的通信代价可以估计为其源顶点的热度。因此，边缘(u,v)的传播成本估算如下。

$$com_{u,v}=h_u$$

值得注意的是，顶点热度和边缘热度的估计方法是启发式的，可以根据算法的特点自定义。也可以在计算过程中执行动态图划分或动态工作负载平衡，因为我们可以在计算过程中获得精确的顶点热度。但我们的初步结果显示，动态负载均衡的性能不理想，因为热值不稳定，会带来较大的迁移成本。在本文中，将重点放在静态热估计上，而将动态图划分作为未来的工作。

（1）节点分配相同数量的顶点热度

在具有优先级调度的异步框架中，顶点越热，它们需要的计算资源就越多。我们应该根据顶点热度来分配计算资源。假设在一个同构集群中，节点具有相似的计算能力，为了全局提高工作效率，应该平衡分区之间的热度，使每个分区分配相同数量的计算资源。因此，划分方案的第一个目标是为节点分配相同数量的顶点热度。

在上面的分析中有一个假设，即使用全局优先级调度器。然而在实践中，全局调度器在大规模集群中是非常昂贵的。使用一个运行在每个 worker 上的局部调度器来模拟全局调度器，也就是说，优先级调度是并行工作的，并且是基于每个 worker 的，这样就避免了昂贵的全局协调开销。例如在图 1 中，红色顶点具有较高的热度值，而蓝色顶点具有较低的热度值。假设每一轮都选择两个最热的顶点进行计算。如图 1c 所示，因为只有一个热顶点被分配给 worker 2(其他的是冷顶点)，所以至少可以安排一个冷顶点。在计算资源有限的情况下，这会降低 worker2 的计算效率。

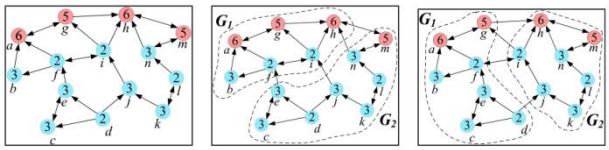


图 1 基于热度的分区例子

给定一个热值集H，有它的累积分布函数$F(x)=P(X<x)$，在优先级调度下，具有高热值的顶点子集H_0很可能被选择优先执行。设其与整个集合的比例为$n\%=\frac{|H'|}{|H|}$，通过分布式计算，将H划分为k个不相交的分区，$H_1,H_2,\ldots,H_k$，其中$H_1\cup H_2\cup\ldots\cup H_k=H$且$H_1\cap H_2\cap\ldots\cap H_k=0$。从每个分区中选取优先级最高的$n\%$元素，得到$k$个候选子集$H_1$，$H_2$，...，$H_k$。如果每个 worker 上的局部热值分布与全局热值分布一致，则局部优先级调度可以达到与全局优先级调度相同的效果。

（2）最小化每个分区与原始图之间的 HJS 距离

由于箱的数量也影响着密度估计。更大数量的容器(更宽的容器)可以提高密度估计的精度，但可能会由于抽样随机性而增加噪声。没有“最好”的箱子数量。因此，在箱的数量固定的情况下，第二个目标是最小化每个分区与原始图之间的 HJS 距离。

（3）最小化分区之间的通信成本

在分布式优先级迭代计算中，网络通信对分布式计算的性能影响很大。在发送重要消息时，网络流量过大可能导致消息阻塞，这些消息有助于加速优先级计算。因此，第三个目标是最小化分区之间的通信成本。

综上所述，分布式优先图计算量身定制的热度平衡分区定义如下：

即给定图 $G=(V,E,H)$，通过分区数 k、热平衡分区，来寻找分区方案$\mathcal{G}=\{G_1,\dots G_k\}$

(1) 使得每个分区的热值之和是平衡的。

$$\min\sum_{i=1}^{k}|\sum_{v\in V_i}h_v-\frac{\sum_{v\in V}h_v}{k}$$

(2) 使得各分区与原始图之间的热度分布的方差最小

$$\min\sum_{i=1}^{k}HJS(P||P_i)$$

(3) 沟通被最小化

$$\sum_{i=1}^{k}Com_i$$

其中$Com_i=\sum_{u\in V_i,v\notin V_i}(com_{u,v}+com_{v,u})$

3.1.2 启发式的基于流的图划分算法 SPb-HBP

作为 NP-hard 问题，很难找到同时满足这三个要求的最优划分方案。作为一个多目标优化问题，使用一些基于局部搜索的方法来寻找局部最优解也是非常昂贵的。因此提出了一种高效的基于流的启发式算法 SPb-HBP。

在 HBP 中想要实现三个目标，如果每个箱子都用平衡的热值进行划分，可以同时实现第一个和第二个目标，即

$$\sum_{i=1}^{k}|\sum_{v\in H_{ji}}h_v-\frac{\sum_{v\in H_j}h_v}{k}|$$

最终分区目标为：

（1）平衡 bin 分区的热度

（2）最小化通信成本

称这种 HBP 的变体为 Per-Bin 热平衡分区(Pb-HBP)。图 2 也描述了 Pb-HBP 的思想。将不同热级别容器中的顶点分别进行分区，同时最小化分区之间的通信开销。

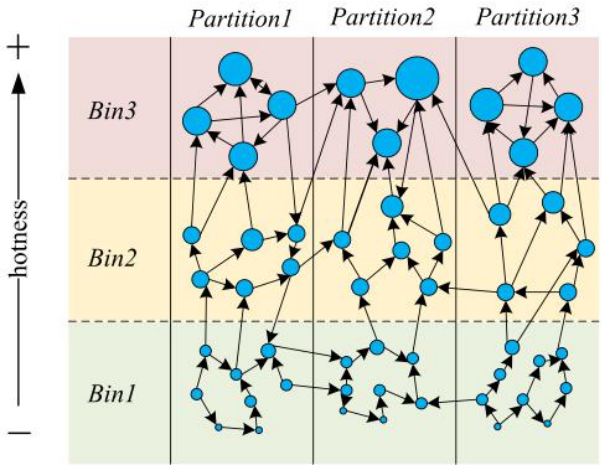


图 2 Pb-HBP 图分区

3.1.3 分布式 SPB-HBP

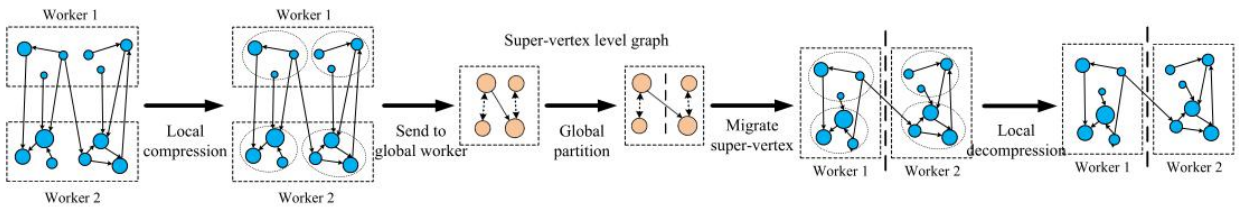
当图数据非常大时，图数据可能是分布存储或分布采集的，即图数据最初存储在多个 worker 上。为了利用 SPb-HBP 对图进行分区，需要一种分布式分区算法。但分布式图划分算法需要大量的 shuffle 通信来交换顶点数据和边缘数据。为了了解邻域信息，我们可能需要进行几轮数据洗牌和同步，这可能会导致大量的网络流量和性能下降。

本文提供的思路如下：

每个分布式工作器将其分配的图数据压缩成一个紧凑和信息丰富的图摘要(结合顶点和边)，揭示原始图的底层拓扑特征和顶点/边热度。然后将这些局部图摘要发送到一个全局工作者，在那里对这些局部摘要执行 SPb-HBP 以获得全局分区结果。根据图摘要的全局结果，合并的顶点可以从一个工作者迁移到另一个工作者，在那里它们被解压到原始的顶点。由于只有图摘要在 wokers 之间进行变换，因此可以大大节省通信成本。

在图 3 中展示了分布式图的分区过程，它包含三个阶段，分别是局部压缩、全局分区和局部解压缩。

图 3 分布式 SPb-HBP 图分区概述



3.2 Geo-Cut划分

3.2.1 Geo-Cut 划分思想

考虑图$G(V,E)$，输入数据存储在M个地理分布数据中心，其中V是图中顶点集，E是图中的边集，每个顶点$v(v=0,1,\ldots,|V|-1)$有初始位置$L_v(L_v \in [0,1,\ldots,M-1])$，是顶点$v$的输入数据存储的地方。

在分布式 GAS 模型中，DC 间网络流量主要来自汇聚阶段和应用阶段。对于给定迭代我和一个顶点v,每个镜子直流r发送聚合数据的大小$g_v^r(i)$到主收集阶段和主发送的数据大小$a_v(i)$每个镜子更新顶点数据在应用阶段。为了简化数据传输时间的计算，假设在收集阶段和应用阶段之间存在一个全局障碍。因此，迭代i的 DC 间数据传输时间可以表示为收集阶段和应用阶段数据传输时间的总和。在每个 DC 中，当数据在上行和下行链路上都完成时，数据传输也就完成了。因此,有:

$$T(i)=T_G(i)+T_A(i)=\max_r T_G^r(i)+\max_r T_A^r(i)$$

$$T_G^r(i)=max(\frac{\sum_v I_v^r\cdot\sum_{k\in R_{v,k\neq r}} g_v^k(i)}{D_r},\frac{\sum_v(1-I_v^r)\cdot g_v^r(i)}{U_r})$$

$$T_A^r(i)=max(\frac{\sum_v I_v^r\cdot a_v(i)\cdot(|R_V|-1)}{U_r},\frac{\sum_v(1-I_v^r)\cdot a_v^r(i)}{D_r})$$

其中I_v^r是一个布尔显示器，显示 DCr中顶点v的副本是否为主节点（I_v^r=1 或 0），R_v是包含至少一个v副本的 DC 集合，最初只包含L_v，U_r和D_r分别为 DCr的上行带宽和下行带宽。

数据中心间的数据通信开销为数据采集和应用阶段的数据上传开销之和。表示将数据从 DCr上传到 Internet 的单位价格为P_r，有：

$$C_{comm}(i)=\sum_v\sum_{r\in R_v} P_r\cdot[I_v^r\cdot dm_v+(1-I_v^r)\cdot g_v^r(i)]$$

其中dm_v等于$a_v(i)\cdot(|R_V|-1)$，表示表示v的主副本上传的数据量。

基于上述分析，可以将地理分布图划分问题表述为约束优化问题：

$$minT(i)$$

$$C_{comm}(i)\leq B$$

3.2.2 地理感知图划分

该算法包含两个优化阶段，以减少优化开销。在第一阶段，研究了图处理过程中的流量模式，提出了一种考虑网络价格异质性的流启发式算法，以最小化数据中心间的数据传输代价。在第二阶段，提出了两种考虑网络带宽异构性的分区优化启发式算法，在不违反预算约束的情况下减少数据中心间的数据传输时间。

（1）成本感知流图划分

在确定了顶点的初始位置(即顶点输入数据的位置)后，采用流图划分方法对图进行快速划分。把一个图看成是边流$e_0,\ldots e_{|E|-1}$，分配给一个图形分区，分区个数与数据中心个数相同。集合中的边的顺序决定了流分区方法首先分配的边，本文采用随机地对流中的边进行排序。在流划分中，一个重要的设计参数是启发式，它决定在哪里放置进入的边。

（2）Performance-Aware 分区优化

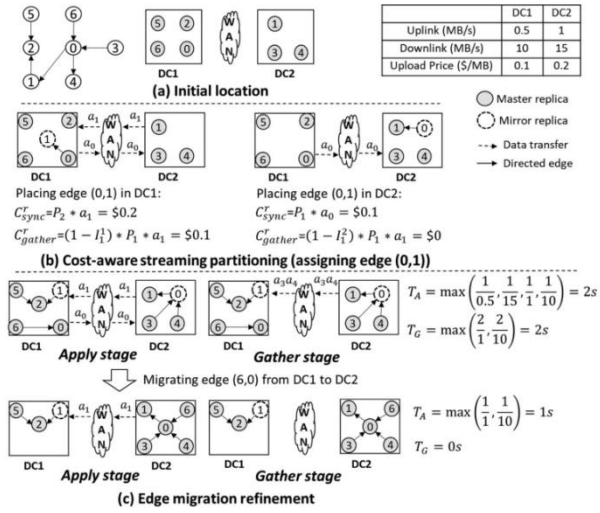
在此阶段，在成本感知划分的基础上，考虑地域分布数据中心网络带宽的高度异构性，提出了两种启发式方法，在满足预算约束的情况下，降低数据中心间的数据传输时间。首先，我们提出了一种简单而有效的分区映射启发式算法，该算法在满足预算约束的情况下，反复尝试切换分区映射以减少数据中心间的数据传输时间。其次，考虑网络带宽的异构性，采用边缘迁移的方式减小瓶颈数据中心的数据流量，在满足预算约束的情况下，进一步减少数据中心间的数据传输时间。

文章中给出了一个简单的图划分示例：利用 Geo-Cut 分割一个有 7 个顶点和 6 条边的小图。

顶点的初始位置如图 4a 所示。首先，采用成本感知的流图划分方法，得到一个初始的划分方案。假设第一个要赋值的边是边(0,1)，在图 4b 中展示了如何放置它当 R0 和 R1 不相交时，可以计算出将边放置在 DC1 和 DC2 中所增加的总成本分别为 \$0.3 和\$0.1。因此,边(0,1)应放在 DC2 中。用相同的方法放置所有边后，得到图的初始分区解。

接下来应用分区映射细化技术。在本例中，切换两个分区的映射不会带来任何好处，所以分区解决方案保持不变。

最后确定了数据中心间数据通信的瓶颈，并应用边缘迁移优化。计算出申请和收集阶段的时间都是 2，可以确定 DC1 的上行链路是申请阶段的瓶颈。为了解决这个瓶颈，根据数据大小对顶点 0、2、5 和 6 进行排序。由于所有的顶点都有相同的数据大小，选择删除顶点 0 并迁移边(6,0)从 DC1 到 DC2。边缘迁移后，应用阶段和收集阶段的时间分别减少到 1s 和 0s。



	DC1	DC2
Uplink (MB/s)	0.5	1
Downlink (MB/s)	10	15
Upload Price ($/MB)	0.1	0.2

图 4 一个应用 Geo-Cut 两阶段优化方法对图进行划分的具体实例

3.2.3 分区动态图

使用 Geo-Cut 划分也能够以较低的开销自适应地划分动态图。添加和删除一个顶点可以表示为添加和删除连接到这个顶点的边，因此将动态图的更改抽象为边插入和边删除，假设一个图没有孤立的顶点。采用 Geo-Cut 的流式图划分技术，将插入的边直接分配给数据中心。挑战在于，插入/删除的边会改变顶点的数据流量，从而降低现有分区的有效性。例如，当在原低度顶点 v 上插入大量边时，最好将 v 移动到带宽较高的 DC 上。

为了解决这一问题，需要定期将分区细化技术应用于更新的图分区。具体来说，是使用分区映射来重新定位插入了新边的图分区。采用边缘迁移技术，将边缘从当前的瓶颈数据中心迁移到已删除边缘的数据中心，以减少数据中心间的数据传输时间。每次插入/删除边时执行细分的代价很高。因此，需要决定分区细化的时机。具体来说，使用与流图分区相同的方法计算边缘插入/删除对分区造成的数据流量大小的增加/减少。每当边缘插入/删除导致的数据流量更改的数量大于阈值时，我们就会触发分区细化。默认情况下，我们将阈值设置为分区总体数据流量大小的 10%。插入一组边缘 E'的时间复杂度是 $O(|E'| + MaxIter \times R)$，删除一组边的值是 $O(L_Q \times C \times R)$，其中 R 是应用的分区细化操作的次数。这比重新划分整个图要快得多。

3.3 EBV划分

3.3.1 EBV 划分思想

（1）以子图为中心的批量同步并行模型与工作流

利用以子图为中心的批量同步并行(BSP)模型测试的所有划分算法。该模型将整个图划分为若干个子图。每个子图绑定到一个 worker，并且每个 worker 只处理一个子图。以子图为中心的 BSP 模型的整个图处理过程是迭代的，可以分为三个超步骤:计算阶段(更新图)、通信阶段(交换消息)和同步阶段(等待其他 worker 完成消息交换)。在计算阶段，顺序算法以当前子图和接收消息为输入。每个子图通过顺序算法更新。在通信阶段，只允许复制顶点之间的消息发送/接收操作。通常，消息包含足够的信息来更新接收顶点的状态。同步级被设计成一个隔离超级台阶的屏障。在这个阶段，每个 worker 等待其他 worker 完成它们的计算和通信阶段，也就是“批量同步”。

（2）高效和平衡的顶点分割

EBV 算法以图 G(V, E)和子图的个数 p 作为划分结果的输入和输出，keep、e_{count}和V_{count}作为辅助变量，进行动态更新，$keep[i]$保存第i个子图的顶点集，e_{count}和V_{count}表示第i个子图已分配的边数和定点数，由此抽象出一个估计函数

$$Eva_{(u,v)}(i) = \prod(u \notin keep[i]) + \prod(v \notin keep[i]) + \alpha\frac{e_{count}[i]}{|E|/p} + \beta\frac{v_{count}[i]}{|V|/p}$$

用估计函数$Eva_{(u,v)}(i)$来衡量将边(u,v)赋值给子图i的效益。$Eva_{(u,v)}(i)$越小，说明该赋值越合适。我们的算法会将边(u,v)分配给 worker i，它的 Eva(u,v)(i)最小。利用超参数 α 和 β 来调整评价函数中边缘和顶点平衡的灵敏度。它们越大，我们的算法就越注重边缘和顶点的平衡。

（3）边和顶点不平衡因子的上界

在此阶段证明了一般图的 EBV 的边不平衡因子和顶点不平衡因子的最坏情况上界分别为 $1+\frac{p-1}{|E|}(1+\frac{2|E|}{\alpha p}+\frac{\beta}{\alpha}|E|)$和 $1+\frac{p-1}{\sum_{j=1}^{p}|V_j|}(1+\frac{2|V|}{\beta p}+\frac{\alpha}{\beta}|V|)$，这也意味着这意味着我们可以通过调整超参数 α 和 β 来限制边和顶点不平衡因子的上界。

对 EBV、Ginger、DBH、CVC、NE 和 METIS 的执行时间进行了比较。为了理解不同划分算法性能背后的机制，研究了单个子图的计算和通信模式。选择 CC 和 4 个 workers 在 LiveJournal 上作为一个代表性的例子。基于子图为中心的 BSP 编程模型及其工作流程，将超步划分为三个阶段:计算阶

段、通信阶段和同步阶段。将计算时间和通信时间分别记录为$comp_i^k$和$comm_i^k$,其中i表示子图 ID，k表示第k个超步，$comp$和$comm$是所有 workers 的平均计算时间和通信时间，因此他们被计算为$\sum_{i=1}^{p}\sum_{k=1}^{s} comp_i^k/p$和$\sum_{i=1}^{p}\sum_{k=1}^{s} comm_i^k/p$,其中$p$和$s$为 workers 和超步的总数。定义$\Delta C^k$为$\max(comp_i^k + comm_i^k) - \min(comp_i^k + comm_i^k)$，表示第$k$个超步中最长的同步等待时间，因此使用$\Delta C = \sum_{k=1}^{s} \Delta C^k$作为工作负载平衡的度量。

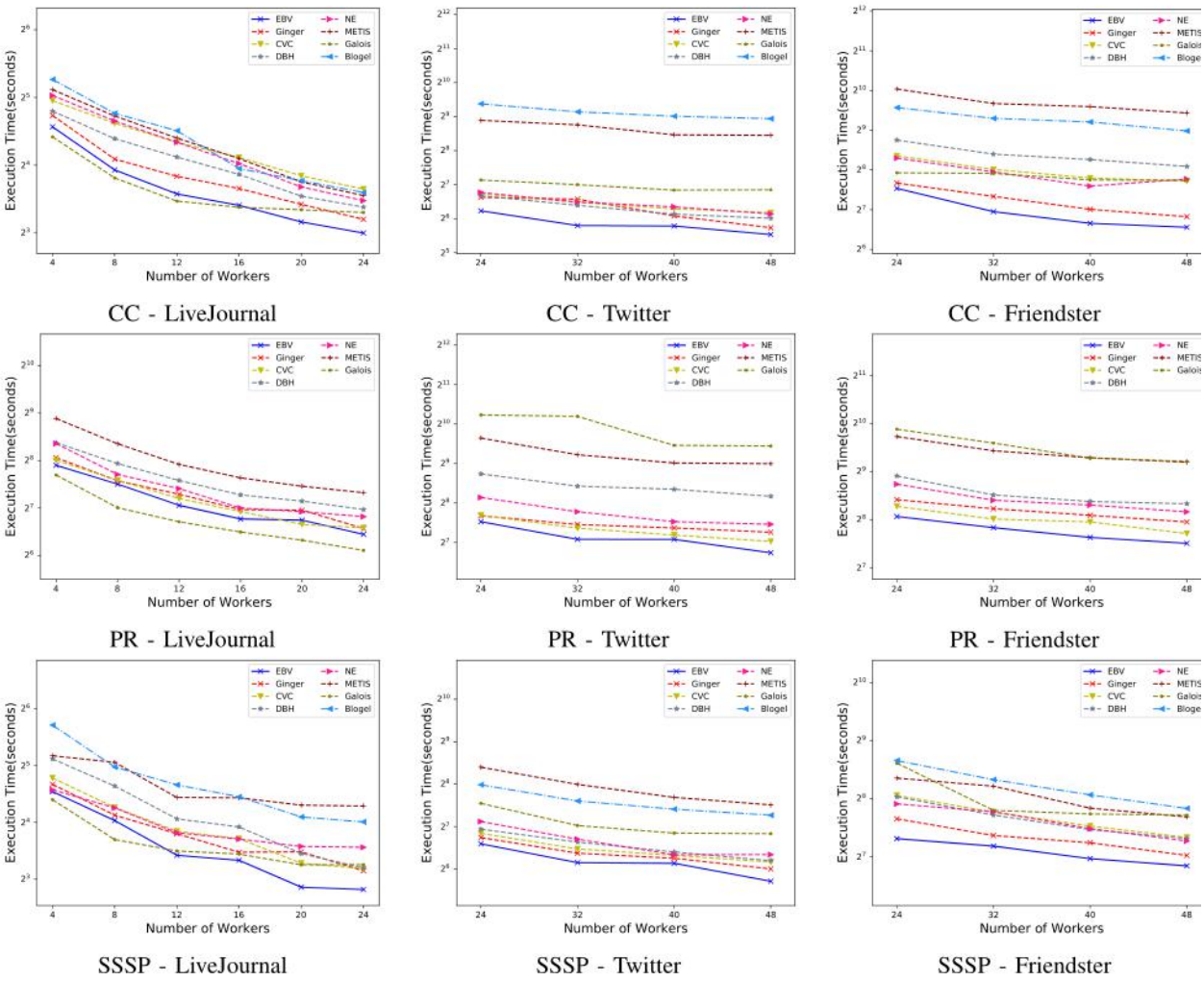
3.3.2　实验分析

文章对 EBV、Ginger、DBH、CVC、NE 和 METIS 的执行时间进行了比较。还将这些结果与最先进的框架 Galois 和 Blogel 进行了多样性比较。在这个比较中，分区开销不包括在内。图 5 显示了性能比较。

值得注意的是，Blogel 使用了一个基于多源的 Graph Voronoi Diagram 分区器来检测在分区阶段连接的组件，确保每个块都是“连接的”。在随后的 CC 计算阶段，它只将较小的连接组件合并为较大的组件，而不执行顶点级计算。为了进行公平的比较，我们将分区阶段的预计算时间添加到 Blogel 的 CC 总时间中。Blogel 也被排除在 PR 比较之外，因为它的 PR 实施并不标准，其结果也没有直接可比性。感兴趣的读者可以参考[19]了解更多细节。从图 5 可以看出，在大多数情况下，EBV 的表现最好。与 Ginger、DBH、CVC、NE 和 METIS 相比，其执行时间平均分别减少了 16.8%、37.3%、25.4%、31.5%和 63.0%。尽管 Galois 在 PR-LiveJournal 上表现良好，甚至优于 EBV，但对于更大的图表来说，它是有限的。

文章还比较了 CC 和 SSSP 不同框架在非幂律图(USARoad)上的性能。图 6 显示了 CC 和 SSSP 的性能比较。在本例中，METIS 的性能可与 EBV、Ginger 和 CVC 相媲美，但与图 5 不同。在所有划分算法中，NE 算法的性能最好。基于这些实验，我们可以得出结论，EBV 在幂律图上优于其他划分算法，在非幂律图上接近 METIS

图 5 CC、PR 和 SSSP 在不同图形上的跨系统性能比较



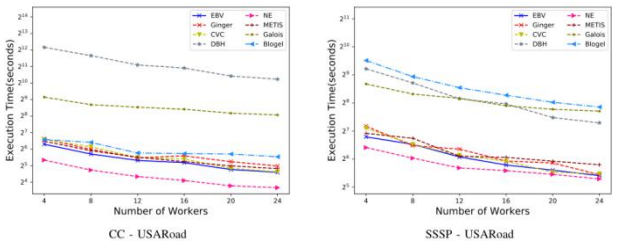


图 6 CC 和 SSSP 在 USARoad 的比较

4 总结与展望

社交网络、网页链接、城市道路交通网和大规模集成电路等均可抽象为顶点和边的数量在千万级以上的大规模图（数据），优良的大规模图划分结果可以充分利用集群资源、提升数据处理效率。已有许多研究者以线性规划、多级划分、启发式等方法对大规模图划分问题进行了研究，并提出了许多求解算法。但已有算法存在划分效率低、未充分考虑图结构、划分结果质量差等问题,因此图划分问题的研究也随着实际应用的需求不断驱动。

文章 1 针对传统 k-平衡图划分无法在优先级异步迭代框架中工作，提出了一种新的图划分思想——热值平衡划分，专门针对优先顺序的调度框架，提出单次算法，同时也提出分布式版本。

文章 2 针对许多图应用都部署在地理分布是数据中心，提出了一种地理感知的图形划分方法 Geo-Cut,满足跨数据中心组王成本预算约束的前提下，最小化图处理的数据传输时间。

文章 3 针对以子图为中心的模型，提出了提高具有子图中心模型的框架在大规模幂律图中的性能。

参 考 文 献

[1] S. Gong, Y. Zhang and G. Yu, "Accelerating Large-Scale Prioritized Graph Computations by Hotness Balanced Partition," in IEEE Transactions on Parallel and Distributed Systems, vol. 32, no. 4, pp. 746-759, 1 April 2021, doi: 10.1109/TPDS.2020.3032709.

[2] A. C. Zhou, B. Shen, Y. Xiao, S. Ibrahim and B. He, "Cost-Aware Partitioning for Efficient Large Graph Processing in Geo-Distributed Datacenters," in IEEE Transactions on Parallel and Distributed Systems, vol. 31, no. 7, pp. 1707-1723, 1 July 2020, doi: 10.1109/TPDS.2019.2955494.

[3] An_Efficient_and_Balanced_Graph_Partition_Algorithm_for_the_Subgraph-Centric_Programming_Model_on_Large-scale_Power-law_Graphs