DisHAP:基于层次亲和聚类的分布式大图划分算法

柳 菁,李 琪
(绍兴文理学院计算机科学与工程系,浙江绍兴 312000)

摘 要: 平衡图划分是改善并行图计算性能的关键.一个良好的划分算法应保证划分后的子图在负载均衡的前提下,减少子图之间的交互边(切割边)规模,从而减少网络通信.对此,本文设计一种基于层次亲和聚类的分布式大图划分算法(DisHAP).该算法采用亲和聚类的思想,将图初始划分为规模相等的k个子图;再将结果映射成顶点序列,以线性嵌入顺序处理节点,通过局部交换策略优化割边率;最后将DisHAP应用在MapReduce框架中,使用多种真实及理论图数据,与现有的大图划分算法做比较分析.以Twitter图为例,划分2,4,8,16,32个子区,相较于现有的大图划分算法(LDG,BLP,Spinner,Fennel,ParMetis及PSA-MIR算法),割边率减少1.7%~30.2%,说明了该算法的优越性.同时该算法具有良好的可扩展性,划分的子区数量及图的规模对划分时间具有较低的影响.

关键词: 分布式大图划分;层次聚类;局部优化;分布式图计算;平衡划分
中图分类号: TP311 **文献标识码:** A **文章编号:** 0372-2112(2021)10-2002-10
电子学报URL: http://www.ejournal.org.cn **DOI:** 10.12263/DZXB.20190919

DisHAP: A Distributed Partition Algorithm for Large Scale Graphs Based on Hierarchical Affinity Clustering

LIU Jing, LI Qi
(*Department of Computer Science and Engineering, Shaoxing University, Shaoxing, Zhejiang* 312000, *China*)

Abstract: It is the key to improve the performance of graph calculation by improving the efficiency of the graph partitioning algorithm and reducing the communication edge scale between the subgraphs. Due to the limited memory capacity of single computing node, it is difficult to meet the partitioning requirements of large-scale graphs in terms of partitioning efficiency and partitioning quality. In this paper, a distributed graph partitioning algorithm based on hierarchical affinity clustering for massive scale graphs is designed, called DisHAP. It uses the Boruvka minimum spanning tree algorithm to balance cluster the graph under the constraint condition of the input graph according to the vertex similarity, and partitions the graph into k subgraph (partitions) with equal size. In order to optimize the fractiom of edges cut between these large subgraphs, we map the initial partition results to the vertex sequence and cut into a large number of subsections, randomly select the sub slice pairs in the neighbor subgraph sequence, and migrate the vertices according to the mutual exchange and the single vertex positive profit. Thus, the optimization problem of large data volume is converted to a large number of small problems to solve. The algorithm is applied to the MapReduce framework to effectively improve the efficiency of the algorithm. Finally, we use various actual and theoretical graph data to compare with existing graph partitioning algorithms to verify the effectiveness of the DisHAP algorithm. Taking the graph Twitter as an example, it is partitioned into 2,4,8,16, 32 partitions. Compared to LDG, BLP, Spinner, Fennel, ParMetis and PSA-MIR algorithms, the fraction of edges cut is reduced by 30.2%, 29.4%, 10.2%, 7.8%, 1.7%, and 3.3% respectively.

Key words: distributed large-scale graph partitioning; hierarchical clustering; local optimization; distributed graph computation; balanced partitioning

1 引言

图的k-划分是NP难问题[1],广泛应用于数据挖掘[2]、VLSI设计[3]等领域.从20世纪90年代初期至今,国内外研究者不断对图划分及其相关问题进行深入研

收稿日期:2019-08-12;修回日期:2021-07-26;责任编辑:孙瑶
基金项目:国家自然科学基金青年科学基金(No.62002226)

究,提出了许多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.

(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. 离线划分主要包括普划分策略、启发式划分策略及混合划分策略.

①谱划分方法最先由Donath和Fiedler提出[4],因为要使用矩阵对应的特征值,所以称为谱划分方法. 利用谱方法进行图划分时,需要先求出图的拉普拉斯矩阵,然后求出该矩阵的所有特征值及对应的特征向量,选取次小特征值以及对应的特征向量.

②K/L算法[5]是由Kernighan和Lin提出的最早的启发式算法之一,通过对目标函数的迭代优化最终得到近似最优解,对于只有上万顶点的图,也很难处理. 关于启发式划分的相关方法研究比较多,如基于模拟退火划分算法[2]、基于蚁群划分算法[6]、基于Tabu搜索划分算法[7]等.

③混合方法[8]是将多种划分方法结合起来使用以便提高划分质量. 比如,可以先使用某种划分方法得到一个初始划分,在初始划分的基础上再利用局部算法(KL/FM)优化划分结果. 由于较高的时间复杂度,离线划分很难应用在大规模图数据划分中.

(2)流式划分

流式划分策略是在数据加载过程中对图数据加载边划分,其假定数据以节点流或者边流的方式到达. 在划分过程中,按照已到达数据的分布信息,通过一组启发式的规则来决定当前到达数据的划分位置. 与离线划分相比,流式划分方法只需要对数据扫描一遍即可实现数据划分,极大地提高了划分效率. 但是因为在划分过程中只能够依据部分数据来决定当前顶点的划分位置,牺牲了部分划分精度来换取划分效率. 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他的在线联系人,如果他的大部分好友与他不在同一个服务器上,他们之间将通过不同网络基础设施进行通信,通信量明显增多,平衡流算法很好地解决了这一问题,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降低. Awadelkarim等人[9]提出了"restreaming partitioning"方法,即相同的图数据被反复加载、处理时(例如,第1次加载用于计算单源最短路径,第2次加载用于计算PageRank),可以利用上一次流式划分的结果来改善本次划分的效果. 实验表明,该机制可以提供与Metis近似的划分效果.

这些算法通常是集中式算法,便于维护复杂的启发式规则,保证相对较好的划分效果,但是其扩展性显然受到单计算节点处理能力的限制,且面对大规模的图划分,不论是离线算法还是流式算法,由于单计算节点内存容量的限制,必然会降低计算性能. 分布式框架的发展,使分布式资源更容易访问. 以Google提出的MapReduce为例,MapReduce的并行计算模型主要是针对海量数据的处理,它在底层对数据分割、任务分配、并行处理、容错等细节问题进行封装,极大地简化了并行程序设计. 在使用MapReduce进行并行计算开发时,用户只需注意到自身要解决的并行计算任务. 因此,本文中设计了一种基于层次亲和聚类的分布式大图划分算法(DisHAP),通过分布式处理提升大图划分效率,有效解决了大规模图的划分问题.

本文的主要贡献如下.

(1)设计了基于Boruvka算法层次亲和平衡图聚类作为初始划分. 以顶点相似度作为距离度量,迭代合并距离较近的两类顶点,并移去子图中邻点相似度和值最小的顶点以约束规模过大的子图. 在没有后续优化的情况下,划分质量也接近于现有的某些大图划分方法,如Spinner算法[10]、线性权重贪心流算法(Linear Weighted Deterministic Greedy, LDG)[11]及平衡标签传播划分算法(Balanced Label Propagation, BLP)[12].

(2)针对大规模子图之间的割边率优化问题,设计了降维的操作,通过将初始划分的结果映射为顶点序列并切分为一定数量的子片,随机选择相邻子图中的两个子片,根据互交换正收益及单点正收益迁移顶点,以此达到优化割边率的目的.

(3)实现了MapReduce并行编程模式下的DisHAP算法,通过分布式处理提升划分的效率. 最后在Hadoop平台上搭建的集群环境中进行实验,并与其他现有大图划分算法做对比,实验结果验证了DisHAP算法的有效性.

2　相关工作

由于分布式系统的协同高效性且分布式资源获取变得越来越方便,研究者们逐渐从单计算节点的图划分工作转为并行图划分的研究. 最典型的是多层次算法Metis[13]及其并行版本ParMetis[14],该算法包括三个执行步骤:粗化、初始划分和细化.

(1)粗化阶段. 通过将输入图的某些节点聚合成一个节点来逐步构建一系列的缩并图,每次生成的缩并图作为下一级缩并过程的输入,最终得到一个规模足够小的图,其节点数与边数都较少.

(2)初始划分阶段. 在粗化过程最终生成的图上,

选择现有的某种图划分算法计算k路划分,由于该图规模较小,此过程较快.

(3)细化阶段. 将划分后的缩并图沿着缩并阶段的逆过程,逐步还原到原始输入图,由于基于粗化图的划分结果不一定是原始图的最佳划分,所以将粗化图逆向映射回原图过程中的每一级还原的同时,利用KL/FM等启发式算法对割边率进行局部优化,直至最终得到原始图的一个划分. 该算法广泛应用在各类图的划分,对于百万规模以内的图,通常具有较好的实际效果,与之类似的算法还包括Chaco[6]和Scotch[6]. 许多研究者针对Metis的三个主要步骤分别进行了改进,文献[2]中,作者在Metis粗化阶段采用标签传播算法对大图进行压缩,这样将具有社区结构的顶点归为一类,在提高划分效率的基础上提高了划分质量. 在文献[15]中,作者将进化算法应用在初始划分及细化阶段,提高划分质量. 类似的方法还有MITS[16]算法.

还有一些研究关于元启发式的并行图划分方法,Moon等人[12]使用基于社区挖掘领域的标签传播算法来解决大图划分问题,称为BLP算法,通过顶点转移再定位,转移那些增益较大的节点. 它将一个最大凹优化问题转化为线性规划问题,既保证了分区平衡,又保证了边的局部性,但线性规划的时间复杂度高,每次迭代都需要解线性规划问题. Nishimura等人[10]提出的Spinner算法,也是基于标签传播模型的方法,在分布式环境下对大规模图数据进行平衡划分. 类似的还有PuLP算法[17]. 但是在标签传播算法中,当周围相同ID的标签数量相等时,顶点ID的选择具有随机性,因此这些算法会导致划分结果的不稳定. 在文献[18]中,作者设计了PSA-MIR算法,通过并行化模拟退火及Tabu搜索的混合策略对图进行划分,但是对于大图而言,由于搜索空间很大,此方法需要大量的迭代次数,即使在并行环境下,划分效率也不是很理想.

与已有的分布式图划分算法不同,本文所提算法由几个逻辑简单的步骤所组成,可以应用在多种分布式框架中,例如MapReduce. 该方法首先通过亲和聚类将图初始划分为指定的k个子图,然后将初始结果映射为类似希尔伯特曲线的一维顶点序列,通过分块的方式,将原问题转化为多个复杂度较小的优化问题,类似于顶点序列的重排列,而不是像目前大多分布式启发式算法一样直接在原图进行顶点的转移优化. 本文的方法对于大图的划分效果显著.

3 问题描述

给定一个无向图$G=(V,E)$,V和E分别代表图G中的点集与边集. 将点集V按照式(1)的约束规则划分到k个独立的子区中$\Pi=\{\pi_1,\pi_2,\cdots,\pi_k\}$. $V(\pi_i)$表示子区π_i中的点集. $E(\pi_i)$表示子区π_i中的边集(指首末端点都在子区π_i中的边). $|V(\pi_i)|$和$|E(\pi_i)|$分别表示子区π_i中的点数量及边数量.

$$(1-\varepsilon_l)\frac{|V|}{k}\leqslant|V(\pi_i)|\leqslant(1-\varepsilon_u)\frac{|V|}{k} \tag{1}$$

$$\varepsilon=\frac{\max_{i\in[1,k]}|V(\pi_i)|}{\frac{|V|}{k}} \tag{2}$$

符号ε_l和ε_u分别表示允许子区含有最少顶点数量的下界系数与最多顶点数量的上界系数. 负载不均衡一般包括子区含有的顶点数不平衡或者子区含有的边数不平衡,本文主要指顶点数,即将最大负载的子区顶点数量与平均负载的比值作为本文负载不平衡的考量,用符号ε表示. $\varepsilon=1$表示平衡划分.

4 算法整体框架

DisHAP算法由7个部分组成,具体过程如算法1所示.

(1)输入图数据.

(2)采用Boruvka最小生成树算法根据顶点相似度对输入图进行约束条件下的平衡图聚类,将图G划分为规模相等的k个子图,每个子图中的顶点距离较近.

(3)将步骤(2)生成的k个子图映射为一维空间的顶点序列,类似于Hilbert曲线,距离近的两点靠在一起,以此将高维数据优化问题转化为一维顶点序列的排列问题.

(4)将每个子区中的顶点序列划分为一定数量的子片,每个子片中含有较少数量的顶点,以此将大规模的子图数据之间的割边数优化问题转化为含有较少顶点的子片之间的排列组合问题.

(5)随机选择相邻子区中的两个子片,按照互交换正收益交换顶点,并不断地迭代此过程,直到割边数趋于稳定时停止迭代.

(6)在给定的负载不平衡系数下,随机选择相邻子区中的两子片,按照单点正收益迁移顶点,并不断地迭代此过程,直到割边数趋于稳定或者到达指定的迭代次数时停止迭代.

(7)输出划分结果.

算法1　DisHAP(G,k,ε,r)

输入:网络$G=(V,E)$,子区数目k,不平衡系数ε,每个子区的分片数目r

输出:k个子区$\{\pi_1,\pi_2,\cdots,\pi_k\}$

1. **for** all $(u,v)\in E$ **do**
2. $w(u,v)$←计算顶点u和v的相似度
3. Π←HAffbl(G,w,k)
4. G_{line}←Π{将Π中的点映射为顶点序列}
5. **for** i=0 to r **do**
6. $q(i)$←G_{line}{将每个子区的顶点序列划分为r个子片}
7. $q'(i)$←Mutexchange$(G,G_{\text{line}},k,r,\varepsilon)${在平衡条件下使用双点交换局部优化}
8. $q''(i)$←Imbalance$(G,G_{\text{line}},k,r,\varepsilon)${在给定的不平衡系数条件下使用单点迁移策略}
9. **for** all i=0 to r **do**
10. Π'←$q''(i)$
11. **return** $\{\pi_1,\pi_2,\cdots,\pi_k\}$

5　基于层次亲和聚类的分布式大图划分过程

DisHAP算法很适合处理大规模的图数据.在本节中,主要从三个步骤来介绍DisHAP算法,分别为初始划分、互交换平衡优化及单点不平衡迁移.

5.1　初始划分

采用基于Boruvka算法的层次亲和平衡划分(Hierarchical Affinity Balanced Partitioning, HAffbl)对输入图数据进行初始划分.但是现有的分布式系统中大多采用Hash方法作为初始划分.本文分别从三部分描述HAffbl划分的过程,首先介绍Boruvka算法,然后对顶点距离的度量进行说明,最后介绍如何在聚类过程中平衡每个子图的规模.

(1)Boruvka算法

Boruvka算法[19]在1926年被首次提出,目的是寻找最小生成树[20].Boruvka算法的主要思想:开始时每个顶点认为是独立的一棵树,对于每个顶点,找出与其相连且距离最小的顶点进行合并形成一棵较大的树(允许出现重复),然后对于每棵树,找出与其相连的距离最小的树再进行合并,如此迭代直到最后只有一棵树,便是最小生成树.Murtagh等人[21]在1956年实现了并行化Boruvka算法.在Boruvka算法执行过程中,其中对于树与树之间距离(两个组合数据)的计算,本文列出了常用的三种聚类方法,分别为单链聚类(single linkage)[22]、全链聚类(complete linkage)[22]和平均链聚类(average linkage)[22].

(2)距离度量

Boruvka算法将边的权值作为两点之间距离的度量值.本文研究的网络是无权图,如果将有边相连的两点距离看作1,无连接的两点距离看作∞,这将导致划分结果很不稳定,主要原因是与点相连的其他邻点到此点的距离都为1,当选择与此点最近距离的邻点时,会产生随机性,这种随机性导致原本应该在不同子区的两点被划分到一个子区中.目前对于无权图有很多方法度量两点间的距离,例如共同邻居比例[20]或个性化的PageRank[23].本文主要关注共同邻居比,即,两个互为邻居的顶点距离度量规则为两个顶点共有的邻居数与这两个顶点邻居数乘积的比值.用式(3)表示u和v两顶点之间的距离$w(u,v)$为

$$w(u,v)=\frac{\text{commonNeihbors}_{uv}+2}{\sqrt{(D_u+1)\times(D_v+1)}} \tag{3}$$

其中,$\text{commonNeihbors}_{uv}$表示顶点$v$和$u$的公共邻居数;$D_v$和$D_u$分别表示顶点$u$和$v$的度数.两点之间的公共邻居数可以通过计算边的三角形个数来求得.

算法2描述了计算顶点相似度的过程.Map函数的输入为$\langle \text{key},\text{value}\rangle$,该键值对有两种:一种是顶点以及顶点度,以$\langle u|D_u,v|D_v\rangle$的形式给出;另一种是边以及边上三角形的个数,以$\langle (u,v),\text{triangleNumber}_{uv}\rangle$的形式给出(边的三角形个数直接体现了节点之间的共同邻居),在文献[24]中详细介绍了其MapReduce实现过程.

算法2　计算顶点之间相似性

输入:边及其顶点度(u,v),$\langle u|D_u,v|D_v\rangle$,边及其边上三角形个数$\langle (u,v),\text{triangleNumber}_{uv}\rangle$

输出:边及边的相似性$\langle (u,v),w(u,v)\rangle$

1. **Map**$\langle \text{key},\text{value}\rangle$
2. key′←(u,v)
3. **if** value包含度数 **do**
4. value′←(D_u,D_v)
5. **else**
6. value′←$w(u,v)$
7. Emit$\langle \text{key}',\text{value}'\rangle$
8. **Reduce**$\langle \text{key}',\text{values}'\rangle$
9. **for** value∈values′ **do**
10. **if** value包含度数 **do**
11. $\text{commonNeighbors}_{uv}$←$\text{triangleNumber}_{uv}$
12. $w(u,v)=\dfrac{\text{commonNeihbors}_{uv}+2}{\sqrt{(D_u+1)\times(D_v+1)}}$
13. Emit$\langle (u,v),w(u,v)\rangle$

(3)平衡子图规模

均衡图划分的目标是生成k个规模相等的子图,所以当Boruvka算法经过l轮迭代($l>0$)达到k($k>0$)棵树

(子图)则停止迭代. 如果算法执行过程中生成树的数目达不到k(小于或者大于k),则在树的数目大于k的临界值时停止迭代(例如k=5,算法运行中产生树的数目靠近5的有8与4,那么在树数目为8时停止迭代). 将距离最近的两棵树进行合并,使树的数目达到k个为止. 对于k个不平衡的子树,必须采用约束条件使其规模相等. 首先对邻近子图进行定义.

定义1 (近邻子图) HAffbl算法(如算法3所示)执行过程中生成的k个树称之为近邻子图,用符号ψ表示,顶点数量为$\left|V_\psi\right|$,边数量为$\left|E_\psi\right|$.

任意选择ψ中某一顶点v,在子图ψ中存在与其互为邻居的顶点u,这两点的距离最近. 当近邻子图的规模达到平均负载时$\left|V_\psi\right|=\frac{n}{k}$,称之为最大近邻子图,用符号$\psi_{\max}$表示.

定义2 (邻点相似度和值) 对于近邻子图ψ中的某一顶点$v\in V_\psi$,近邻子图中v的邻点$v_1\in V_\psi,\cdots,v_n\in V_\psi$与顶点$v$的距离分别为$w\left(v,v_1\right),\cdots,w\left(v,v_n\right)$. 则此点$v$在$\psi$中的邻点相似度和值为$\text{tatal}'_v=w\left(v,v_1\right)+\cdots+w\left(v,v_n\right)$,公式化为

$$\text{tatal}'_v=\sum_{u\in\psi}w(v,u) \tag{4}$$

算法3 HAffbl(G,w,k)

输入:网络$G=(V,E)$, 邻点的相似度值w, 子区数目k

输出:k个子区$\left\{\pi_1,\pi_2,\cdots,\pi_k\right\}$

1. $C^n\leftarrow\{\{v\}|v\in V\}$ {每个顶点作为一个子图}
2. repeat
3. **for** all $(u,v)\in E$ **do**
4. 选择距离最近的两个子图C^i,C^j
5. $C^i\leftarrow C^i\cup C^v$and $C^n\leftarrow C^n\backslash C^j$
6. $n\leftarrow n-1$
7. **if** $\left|V_{C^i}\right|>\left|V_{\psi_{\max}}\right|$ **do**
8. **repeat**
9. 选择C^i中邻点相似度和值最小的顶点v_i
10. $C^i\leftarrow C^i\backslash v_i$and n=n+1
11. $C^n\leftarrow v_i$
12. **until** $\left|V_{C^i}\right|>\left|V_{\psi_{\max}}\right|$
13. 子图C^i不再接受任何其他子图或顶点
14. **until** $C=\varnothing$
15. **return** $\left\{\pi_1,\pi_2,\cdots,\pi_k\right\}$

5.2 互交换平衡优化

上一节产生的结果虽然保证了均衡划分且保持较好的割边数,但是可以通过顶点转移策略进一步优化割边数. 顶点转移策略包括双点交换操作(互交换)及单点迁移操作,本节主要采用互交换操作优化割边数. 互交换是交换两子区中的顶点使割边数减少. 例如经典的KL(Kernighan-Lin)算法首先对两子区中的顶点按照收益值进行排序,然后选择最大收益值的两顶点进行交换并更新所有点对的收益值,如此迭代,直到割边数不再减少为止. 但是现有顶点转移策略都是以每个子区为单位优化割边数,这种以子区为单位的优化方案难以并行化,所以仍然存在效率低下的问题.

(1)映射降维

对于大规模的图数据,如果直接对高维的网络进行优化,计算量非常的巨大,每次都需要计算所有子区之间两顶点互交换的收益值,因此首先想到的是如何对高维网络降维以便于处理. 网络中最重要的是邻居关系的获取,本文主要关注希尔伯特曲线,虽然此方法在一定程度上损失了图的空间结构,但是它能够很好地捕捉邻近空间的节点. 将5.1节初始划分的聚类结果映射为顶点序列,相似性较大的顶点在序列中距离相隔较近. 已有很多工作使用并行化空间填充的方法解决大规模数据的优化问题,本文中也是采用此方法来解决子图之间的割边率优化问题.

(2)互交换

定义3 (单点收益) 点v从所在子区π_{local}迁移到另一个子区$\pi_i\left(i\in[1,k]\right)$,割边减少的数量称之为收益值,用符号$g\left(v,\pi_i\right)$表示,用图1示例说明(图中只显示了点$v$的邻边). 图1中有3个子区$\left(\pi_1,\pi_2,\pi_3\right)$,点$v$在子区$\pi_2$中(黑色实心圆),$EV\left(v,\pi_i\right)$表示子区$\pi_i$中的顶点与点$v$相连的边数,$EV\left(v,\pi_1\right)=1$,$EV\left(v,\pi_2\right)=2$,$EV\left(v,\pi_3\right)=2$. 相应的收益值为$g\left(v,\pi_1\right)=-1$,$g\left(v,\pi_3\right)=0$,$g$值不仅有正值也有负值,用数学形式表示为

$$g\left(v,\pi_i\right)=EV\left(v,\pi_i\right)-EV\left(v,\pi_{\text{local}}\right) \tag{5}$$

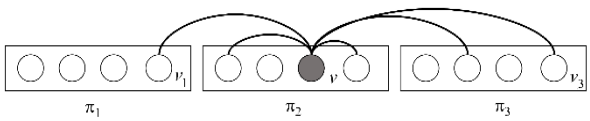

图1 3个分区的图划分

定义4 (互交换收益) 已知两顶点v_i,v_j分别属于不同的分区. 即$v_i\in\pi_i,v_j\in\pi_j$,且$\pi_i\neq\pi_j$. 将两顶点进行交换以割边减少的数量为收益值,用符号$\text{Swap}\left(v_i,v_j\right)$表示. 图1所示,点$v_1$与$v$进行交换所获得的收益值为$\text{Swap}\left(v_1,v\right)=-2$,将点$v_2$与$v$进行交换所获得的收益值为$\text{Swap}\left(v,v_2\right)=-1$. Swap值也是不仅有正值也有负值,用数学形式表示为

$$\mathrm{Swap}\left(v_i,v_j\right)=\begin{cases}g\left(v_i,\pi_j\right)+g\left(v_j,\pi_i\right)-2, & \text{if } \left(v_i,v_j\right)\in E\\ g\left(v_i,\pi_j\right)+g\left(v_j,\pi_i\right), & \text{if } \left(v_i,v_j\right)\notin E\end{cases} \tag{6}$$

为了有效地利用一维顶点序列带来的优势,本节提出了分片操作,将整个序列以子区为单位划分为一定数量的子片,通过交换两子片中的顶点来减少割边数. 对顶点序列分片化处理,将每个子区中的顶点序列均匀划分为r个子片$\left(I_0,I_1,\cdots,I_{r-1}\right)$,随机选择相邻两子区中的子片对(对应于每个Mapper任务),通过交换两子片中的顶点降低割边数,如果互交换两子区中的顶点没有正收益,则重新选择子片对. 分片操作对于划分子区数量少的任务效果更明显,因为子区数越少,相对子区中的顶点数量的规模就越大,完全可以利用分片的操作提高优化效率.

对于两子片中的顶点,采用如下操作:计算两子区中任意两点交换所获得的收益值,按照收益值对顶点对从高到低进行排序;选择收益值最大的且大于零的两顶点进行交换,交换完毕更新这两顶点的邻居节点,迭代计算直到两子区中任意两顶点的互交换收益值小于或等于零. 这种方法是以牺牲可接受的附加成本为代价(需要更多的内存空间)来获得最好的划分质量. 由于子片含有的顶点数相对较少,在并行处理中每对子片都由单独的Mapper任务来处理,所以程序的运行速度相对较快.

5.3　单点不平衡迁移

给定图G的顶点序列G_{line},最小化子图序列之间的割边率,与此问题类似的是最小线性排列(Minimum Linear Arrangement, MinLA),属于NP难问题[6]. 与MinLA不同的是,本节主要讨论的是在指定负载系数下的最小化割边率,即允许子区规模之间一定的不平衡,所以可以利用单点迁移来进一步优化割边数,通过单点正收益来迁移顶点,公式化为

$$\arg\max \sum_{V_{\text{local}}\in \Pi_{\text{local}},\Pi_{\text{local}}\neq\Pi_i} g\left(V_{\text{local}},\Pi_i\right) \tag{7}$$

解决此问题的方法有很多,比如可以通过计算每个顶点的单点收益值进行顶点迁移或者可以通过启发式的一些方法寻找全局或者局部最优解. 为了充分利用步骤2的分片操作方式,本文采用了与互交换平衡优化相类似的操作,具体步骤如下所示.

(1)将顶点序列以子区为单位划分为r个子片,每个子片中含有较少数量的顶点.

(2)随机选择两子区中的两个子片,首先计算这两子片中的顶点迁移到对方子区所获得的收益值,按照收益值对它们进行降序排序. 选择收益值最大的且大于零的顶点进行迁移,迁移完毕更新此顶点的邻居节点,迭代计算,直到无正收益或者在指定的迭代次数下停止运行.

(3)当某一子区达到允许的最大负载时,不再接受其他顶点,按照收益值的排序,从大到小选择此子区中迁移到其他子区的顶点,如果不存在则重新选择子片.

6　实验

实验主要验证本文提出的DisHAP算法的性能以及适用性. 首先介绍实验环境及实验中所使用到的数据集. 实验从4个方面来衡量所提出的DisHAP算法:①对于初始划分,分别对比基于Boruvk算法和基于Kruskal算法的单链、完全链、平均链的6种聚类算法,得出性能最好的初始划分方法,并与分布式框架中常用的Hash初始划分方法做比较;②使用不同的初始划分,分析其对DisHAP算法中第2步骤(互交换平衡优化)的收敛次数及割边率的影响;③将DisHAP算法与现有的比较经典的大图划分算法(BLP[12]、LDG[11]、Spinner[10]、Fennel[24]、PSA-MIR[18]、ParMetis[14])在割边率方面做对比分析,证明其对大图划分的优越性;④分析DisHAp算法的划分时间. 实验中主要用到2个指标,不平衡系数ε及割边率η,计算方式为

$$\varepsilon=\frac{\#\text{子区的最大负载}}{\#\text{平均负载}},\eta=\frac{\#\text{割边数}}{\#\text{总边数}} \tag{8}$$

6.1　实验环境

所有算法均以JAVA语言实现,所有测试是在Hadoop平台上搭建的5台服务器构成的集群环境中进行,即1个master和4个slaver. 节点的配置如下:Intel CPU,16GB的内存,500GB的硬盘,运行在CentOS 7的Linux服务器上.

6.2　数据集

实验选用人工数据集和真实数据集对算法进行了评测,数据的详细信息如表1所示. 真实数据集来源于

表1　网络数据集

Name	Number of vertices	Number of edges	Type
hugebubble-10	18. 3M	27. 5M	Mesh
channel	4. 8M	42. 6M	Mesh
nlpkkt240	27. 9M	373M	Mesh
uk-2002	18. 5M	262M	Social
de126	67. 1M	201M	Mesh
rgg26	67. 1M	575M	Mesh
arabic-2005	22. 7M	553M	Social
Friendster	66M	1. 8B	Social
Twitter	40M	1. 5B	Social
LiveJournal	4. 8M	1. 3B	Social
world-roads	5. 6M	1. 1B	road
PLG18	630K	35M	synthetic
PLG20	2. 8M	135M	synthetic
PLG22	9. 1M	518M	synthetic
PLG24	29. 7M	1. 9B	synthetic
PLG26	115M	7. 9B	synthetic

斯坦福大学的公开数据集(http://snap. stanford. edu/). 其中 word-roads 是地理网络数据,Friendster、Twitter 和 LiveJournal 是社交网络数据. 为了防止零切割边的出现,重边、自环、度为零的点都已经被移去. 本文主要考虑的是连通图,所有图数据在网上已经公开. 其中 PLG 是使用 Python 人工 Web 图生成器(http://pywebgraph. sourceforge. net/)创建规模不等的理论幂律随机图,平均度分别设置为 18、20、22、24、26.

6.3 初始划分

首先使用基于 Boruvk 算法和基于 Kruskal 算法的单链、完全链、平均链的六种聚类(简称为 Haffbl_single、Haffbl_complete、Haffbl_average、Kruskal_single、Kruskal_complete、Kruskal_average)算法对图进行初始划分. 划分对象为交通网络图 World-roads,划分子区数 k=4,18,32,46,60. 结果如图 2 右图所示,在所有结果中,Haffbl_single 算法在指定的所有子区数下划分效果最好;划分效果最差的是 Kruskal_complete 算法;Haffbl_complete 算法划分相对不是很稳定;相对于 Haffbl_complete 算法、Haffbl_average 算法、Kruskal_single 算法、Kruskal_complete 算法、Kruskal_average 算法,Haffbl_single 算法分别提升了 0. 38%、0. 22%、0. 2%、0. 2%、0. 56%、0. 44% 的割边率.

由于分布式框架中大多采用 Hash 作为初始划分,所以我们将 Haffbl_single 方法与 Hash 方法划分进行了比较,图 2 左图为对比结果. 划分对象为社交网络图 Twitter,划分子区数 k =4,18,32,46,60. 随着子区数的提高,两种初始划分方法的割边率都在上升. 但是 Haffbl_single 方法平均要比 Hash 方法减少 45%~53% 的割边率. 子区数为 60 时,Hash 划分的割边率几乎超过了 95%,显然对于后续的优化,计算量将会变大. 虽然 Haffbl_single 方法相对于 Hash 方法的复杂度相对较大,但是可以通过并行处理提高计算效率,在后续实验分析中也证明了这一点.

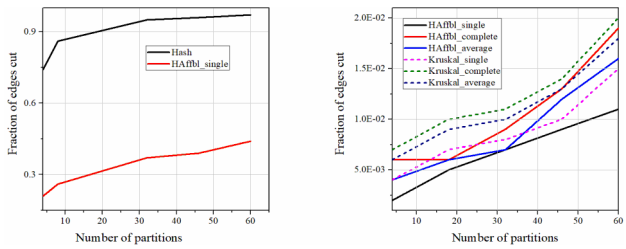

图 2 初始划分之后割边率的对比

6.4 收敛分析

本节主要对 DisHAP 算法的第 2 步骤(互交换平衡优化)的收敛情况做分析. 图 3 显示了在 LiveJournal 上的划分结果,划分分区数 k =2,4,8,32,64. 图 3 左图为分别使用 Hash 方法和 Haffbl_single 方法初始划分方法的划分结果,Haffbl_single 方法划分效果明显好于 Hash 方法,平均要比 Hash 方法减少 32. 33% 的割边率. 图 3 中图显示的是使用互交换平衡优化策略对以 Haffbl_single 方法作为初始划分的结果进行割边率优化(k=2,32). 在 k=2 时,迭代 4~5 轮割边率就趋于稳定. 在 k=32 时,迭代 5~6 轮割边率趋于稳定. 相较于使用 Haffbl_single 方法分别划分 2 个子区和 4 个子区的初始划分结果,互交换平衡优化使割边率分别减少了约 15% 与 13%. 从对不同子区的优化迭代轮次可以看出,划分同一网络图,k 值对迭代次数的影响非常小.

图 3 右图显示的是使用互交换平衡优化策略对 Haffbl_single 方法与 Hash 方法初始划分结果的优化情况(k=2). 相较于 Hash 方法初始划分的结果,使用互交换平衡优化使割边率降低了接近 27%,但是迭代次数较多,迭代 10~15 轮割边率才趋于稳定且优化之后的结果没有在 Haffbl_single 方法初始划分的结果之上优化效果好. 这从侧面反映了 Haffbl_single 方法作为初始划分的优越性.

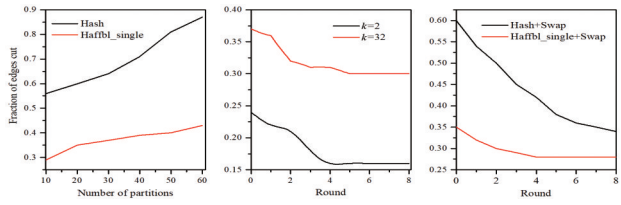

图 3 不同初始划分方法对 LiveJournal 网络划分优化情况

6.5 割边率

本节主要通过与现有的大图划分算法进行对比来衡量所提出方法的割边效果. 表 2 是在图 Twitter 上的测试结果,划分的子区数分别为 k=2,4,8,16,32. 与之对比的是 BLP 算法[12]、Spinner 算法[10]、LDG 算法[11]、Fennel 算法[25]、ParMetis 算法[14]以及 PSA-MIR 算法. 这几种方法已经在相关工作中进行了介绍. 表 2 中 N/A 表示原论文没有给出具体的数值或者由于算法运行时间的过长,导致无法测试出实验结果. LDG 算法与 BLP 算法的划分结果相对较差,与其他算法的划分性能差距比较明显. DisHAP 算法,相较于 LDG 算法平均减少了 30. 2%,最高减少了 35% 的割边率;相较于 BLP 算法,平均减少了 29. 4% 的割边率,最高减少了 37% 割边率. DisHAP 算法和 Fennel 算法表现较好. PSA-MIR 算法迭代时间较长,在本文实验中,子区数量越多,PSA-MIR 算法的运行时间就越长,当划分 16 个子区时,划分时间就超过了 5 个小时,所以文本没有列出其划分结果. DisHAP 算法,相较于 Spinner 算法,平均减少了 10. 2%,最高 12% 的割边率;相较于 Fennel 算法,平均减少了 7. 8%,最高 12% 的割边率;相较于 ParMetis 算法平均减少了 1. 7%,最高 5% 的割边率. 而且从结果中也可以看出,划分子区数量越少,划分效果就越明显.

表2　不同方法对图Twitter划分的结果(割边率)

k	LDG (ε=1.1)	BLP (ε=1.0)	Spinner (ε=1.05)	Fennel (ε=1.09)	ParMetis (ε=1.03)	PSA-MIR (ε=1.0)	DisHAP (ε=1.03)
2	0.31	0.34	0.15	0.07	0.12	0.15	**0.10**
4	0.49	0.55	0.31	0.29	0.24	0.21	**0.19**
8	0.69	0.66	0.49	0.48	0.35	0.40	**0.38**
16	0.80	0.76	0.61	0.59	0.52	N/A	**0.49**
32	0.86	0.80	0.69	0.67	N/A	N/A	**0.58**

6.6　划分时间

划分时间也是衡量性能的一个主要的指标,因此本节对算法的划分时间进行定量分析. 由于ParMetis算法相对于其他对比算法,具有较好的划分质量,所以本文针对ParMetis单独进行了对比. 表3所示为划分k=2的结果. 划分为5次,Avg. cut表示平均割边数. Best cut表示5次中最好的一次割边数. t表示划分时间,单位为s. 由表3可知,当图较小时,DisHAP算法所消耗的时间要多于ParMetis算法,随着图规模的扩大,本文算法的优势逐渐显示出来,主要是因为ParMetis算法在粗糙阶段,过早地停止了压缩,使最终图的规模过大,例如划分图rgg26,最终的压缩图仍然超过了40M. 然后利用启发式算法进行优化,例如利用KL(Kernighan-Lin),时间复杂度为$O(n^2\log n)$. 然后不断地反粗糙也是利用一些启发式方法进行优化. 时间复杂度大大地提高,所以消耗的时间不断地增大. 对于本文所提出的算法,随着图规模不断地扩大,所消耗的时间却不是线性增加,下面也会用实验来证明本文算法的可扩展性.

表3　ParMetis和DisHAP方法对不同图划分的结果(割边数和划分时间)

Graph	ParMetis			DisHAP		
	Avg. cut	Best cut	t/s	Avg. cut	Best cut	t/s
hugebubble-10	1922	1854	4.66	**1910**	1857	8.20
channel	48798	47776	1.55	56912	55959	2.71
de126	18086	17609	23.74	**17002**	**16703**	31.02
rgg26	44747	42739	23.37	**38371**	**37676**	**19.91**
nlpkkt240	1178988	1152935	32.97	1241950	1228086	**25.06**
uk-2002	787391	697767	128.71	**439227**	**390182**	**96.62**
arabic-2005	1078415	968871	1245.57	**561778**	**481141**	**1092.45**

图4左图为DisHAP算法对图Friendster、Twitter和LiveJournal的划分结果,划分的子区数k=2,10,100. 以划分2个子区的时间为基准对照,归一化划分时间,即划分其他子区数的时间与划分2个子区数时间的比值,结果如图4右图所示. 虽然子区数量是指数级往上增长,但是对图的划分时间影响相对不是很大,在三种网络中,即使划分子区数为100,相对于2个子区的划分时间,最多也只是多了接近21%的时间. 因此对于同一个图,划分子区数越多获得的时间受益越大.

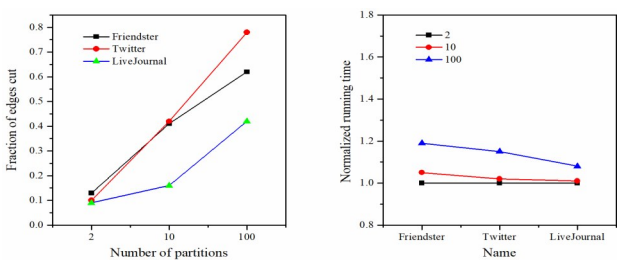

图4　在三种网络图的划分割边率及归一化运行时间

图4右图是对同一个图划分不同子区数的时间进行分析,从其结果中也可以看出图的规模也反映了划分的时间,由于Friendster算法的规模最大,所以算法消耗的时间最多,LiveJournal算法的规模最小,消耗时间也是最少的. 接下来对不同规模网络的划分时间进行具体的实验分析,实验是在理论图上进行的. 以划分图PLG18的时间为基准对照,归一化其他子区的划分时间. 结果如表4所示,随着网络规模成倍的扩大,划分时间并没有显著地增长,因此规模越大的网络获得的时间受益越大,说明了DisHAP算法对大规模网络的有效性. 而对于小规模网络,DisHAP算法的优势会逐渐降低,同时本文所提出的DisHAP算法针对的是无向无权图,而对于有向有权图,需要对邻居之间的相似度度量和割边的优化进行调整. 因为边的权重在某种程度上反映了通信的频度,因此在割边率的后期优化中,需要重新调整优化目标,以最小化子区之间边的总权重为优化目标.

表4 在不同理论图上的归一化运行时间

Graph	|V|	|E|	max degree	running time
PLG18	630K	35M	67K	100%
PLG20	2. 8M	135M	178K	112%
PLG22	9. 1M	518M	402K	131%
PLG24	29. 7M	1. 9B	1. 1M	156%
PLG26	115M	7. 9B	2. 5M	392%

7 结束语

本文设计了一种基于层次亲和聚类的分布式大图划分算法(DisHAP),采用Boruvka最小生成树算法根据顶点相似度对输入图进行约束条件下的平衡图聚类,将图划分为规模相等的k个子图(分区),然后将结果映射为顶点序列. 由此提出分片的概念,将每个子区中的顶点序列分片为数量众多的子窗口,将大数据量的优化问题切分为数量众多的小问题进行解决,随机选择不同分区中的子窗口按照互交换及单点正增益来迁移顶点,进一步优化割边率. 将算法应用在MapReduce框架中,有效地提高了算法的运行效率. 最后,本文使用多种真实及理论图数据,与现有的划分算法做比较分析,验证了DisHAP算法的有效性.

参考文献

[1] Basak A, Li S C, Hu X, et al. Analysis and optimization of the memory hierarchy for graph processing workloads[A]. 2019 IEEE International Symposium on High Performance Computer Architecture (HPCA) [C]. Washington, DC, USA: IEEE, 2019. 373 – 386.

[2] Miller G L, Teng S H, Vavasis S A. A unified geometric approach to graph separators[A]. 1991 Proceedings 32nd Annual Symposium of Foundations of Computer Science [C]. San Juan, PR, USA: IEEE, 1991. 538 – 547.

[3] Stanton I, Kliot G. Streaming graph partitioning for large distributed graphs[A]. Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD 12[C]. New York, USA: ACM Press, 2012. 1222 – 1230.

[4] Stanton I. Streaming balanced graph partitioning algorithms for random graphs[A]. Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms[C]. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2014. 1287 – 1301.

[5] Andreev K, Räcke H. Balanced graph partitioning[A]. Proceedings of the Sixteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures - SPAA04[C]. New York, USA: ACM Press, 2004: 120 – 124.

[6] Liu N, Li D S, Zhang Y M, et al. Large-scale graph processing systems: A survey[J]. Frontiers of Information Technology & Electronic Engineering, 2020, 21(3): 384 – 404.

[7] Benlic U, Hao J K. An effective multilevel tabu search approach for balanced graph partitioning[J]. Computers & Operations Research, 2011, 38(7): 1066 – 1075.

[8] Kokosiński Z, Bała M. Solving graph partitioning problems with parallel metaheuristics[A]. Recent Advances in Computational Optimization[C]. Cham, Germany: Springer, 2018. 89 – 105.

[9] Awadelkarim A, Ugander J. Prioritized restreaming algorithms for balanced graph partitioning[A]. Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining[C]. New York, NY, USA: ACM, 2020. 1877 – 1887.

[10] Nishimura J, Ugander J. Restreaming graph partitioning: Simple versatile algorithms for advanced balancing[A]. Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining [C]. New York, NY, USA: ACM, 2013. 1106 – 1114.

[11] Srinuvasu M A. Analysis of large graph partitioning and frequent subgraph mining on graph data[J]. International Journal of Advanced Computer Research, 2015, 6(7): 12.

[12] Moon B, Jagadish H V, Faloutsos C, et al. Analysis of the clustering properties of the Hilbert space-filling curve[J]. IEEE Transactions on Knowledge and Data Engineering, 2001, 13(1): 124 – 141.

[13] Tsourakakis C, Gkantsidis C, Radunovic B, et al. FENNEL: streaming graph partitioning for massive scale graphs[A]. Proceedings of the 7th ACM International Conference on Web Search and Data Mining[C]. New York, NY, USA: ACM, 2014. 333 – 342.

[14] Battaglino C, Pienta P, Vuduc R. GraSP: distributed streaming graph partitioning[A]. The 1st High Performance Graph Mining Workshop[C]. Arcelona, SPAIN: Barcelona Supercomputing Center, 2015. DOI: 10.5821/hpgm15.3.

[15] Martella C, Logothetis D, Loukas A, et al. Spinner: scalable graph partitioning in the cloud[EB/OL]. https://arxiv.org/abs/1404.3861, 2019.

[16] Pope A S, Tauritz D R, Kent A D. Evolving multi-level graph partitioning algorithms[A]. IEEE Symposium Series on Computational Intelligence (SSCI) [C]. Athens, Greece: IEEE, 2016. 1 – 8.

[17] Tsourakakis C E, Kolountzakis M N, Miller G L. Approximate triangle counting[EB/OL]. https://www.researchgate.net/publication/24356900_Approximate_Triangle_Count -

ing, 2019.
[18] Qian X H. Graph processing and machine learning architectures with emerging memory technologies: A survey [J]. Science China Information Sciences, 2021, 64(6): 1 – 25.
[19] Borůvka, Otakar. O jistém problému minimálním[EB/OL]. https://dml.cz/bitstream/handle/10338.dmlcz/500114/Boruvka_01-0000-6_1.pdf, 2019.
[20] Han M Y, Daudjee K, Ammar K, et al. An experimental comparison of pregel-like graph processing systems[J]. Proceedings of the VLDB Endowment, 2014, 7(12): 1047 – 1058.
[21] Murtagh F, Contreras P. Algorithms for hierarchical clustering: An overview[J]. WIREs Data Mining and Knowledge Discovery, 2012, 2(1): 86 – 97.
[22] Tang C F, Rao Y, Yu H L, et al. Improving knowledge graph completion using soft rules and adversarial learning [J]. Chinese Journal of Electronics, 2021, 30(4): 623 – 633.
[23] Khayyat Z, Awara K, Alonazi A, et al. Mizan: A system for dynamic load balancing in large-scale graph processing[A]. Proceedings of the 8th ACM European Conference on Computer Systems - EuroSys 13[C]. New York, USA: ACM Press, 2013. 169 – 182.
[24] Fiduccia C M, Mattheyses R M. A linear-time heuristic for improving network partitions[A]. 19th Design Automation Conference[C]. Las Vegas, NV, USA: IEEE, 1982. 175 – 181.

作者简介

柳　菁　女,1979年生,浙江绍兴人.2009年于同济大学获计算机工程硕士学位,现任绍兴文理学院计算机科学与技术专业实验师,研究方向为智能信息处理等.

李　琪(通信作者)　男,1987年生,江苏淮安人.2019年毕业于重庆大学获软件工程博士学位,现任绍兴文理学院计算机科学与技术专业讲师,研究方向为复杂网络、知识图谱等.
E-mail:liqi0713@foxmail.com