北京大學

本科生科研中期报告

题目： 面向大数据图计算的在线图划分方法研究

姓　　名：潘成　1300012716

院　　系：信息科学技术学院

专　　业：计算机科学与技术

导师姓名：汪小林

二〇一六 年 三 月

摘 要

随着大数据时代的到来，基于云环境的大图迭代计算已经成为新的研究热点，其中提高图划分算法的执行效率和降低划分后子图之间的通信边规模是改善计算性能的关键。现在已经有的工作主要分成了离线划分和在线划分两个大类，离线算法有些可以得到比较好的通信边规模，而算法的执行效率达不到实时划分的要求；在线算法在效率方面比较不错，但是如何控制通信边的规模又成了一个难题。如何平衡执行效率和划分效果是当前图划分算法一个研究的热点。

同时，由于传统的如 Hadoop 等通用的云平台并不适合迭代式的处理图数据，研究人员基于 BSP 模型提出了新的处理方案，比如 Pregel，Hama，Giraph 等图计算框架。然而，图处理算法需要按照图的拓扑结构频繁交换中间计算结果而导致了巨大的通信开销，这严重影响了基于 BSP 模型的系统的处理性能。本次本科生科研的主要任务就是探求如何在保证效率的情况下，尽可能减少通讯边的开销，并且试图发现不同的图计算需求对划分方案有何影响。

目录

一、进度完成情况

时间	任务	完成情况
2015.5~2015.6	查阅图划分方面的资料，研读相关论文，了解图划分领域研究的成果，逐步理解项目的背景。	完成
2015.7~2015.8	在信息学院网络所系统虚拟化组参与实验室相关课题研究，学习必要的基础知识，了解 **spark** 平台和内存计算的相关知识，了解系统底层结构和实现细节。理解大数据的相关处理方法，**MapReduce**，**Hadoop** 的应用。	完成
2015.9~2016.1	针对大规模图划分问题，学习掌握对划分方案的评价和对划分代价的评估，并且探索研究如何从静态划分出发构建快速在线的划分算法，进行相关实验，提高实验技能并且锻炼动手能力。	完成
2016.3~2016.6	进一步完善在线划分算法的设计，并且付诸代码设计，通过在已有平台的测试进行评估和调整。完成本科生科研中期报告。	待完成
2016.7~2016.8	根据前期获得的经验和实验结果，设计完整的面向大数据在线的图划分算法，进行相关的实验分析并且完成论文的编写。	待完成
2016.9~2016.10 算法	完善面向大数据在线的图划分算法，完成本科生科研结题报告。	待完成

目前完成基础知识的学习和动手实践，接下来的任务是继续深入调研划分算法的实现以及优化。

二、项目概述

随着互联网技术的普及和新兴社交网络等应用的快速发展，大规模图数据的高效处理已经成为学术界和工业界的研究热点。真实图数据集尤其是基于互联网的图数据，其规模通常可以达到数十亿定点和上百亿条边，如 WWW 的 web 页面关联图有 5000 亿定点和一万亿条边；全球最大的社交网站 Facebook 有近 80 亿个顶点和 1040 亿条边。显然，传统的单机串行和小规模集群都无法处理如此庞大的数据。因此，如何高效处理大规模图数据成为亟待解决的问题。

MapReduce 是 Google 推出的一种简单有效的工作流式处理模式，适用于通用的大数据计算，并且现在 MapReduce 的开源实现 Hadoop 已经能够提供商用计算，支持对大数据的批量计算。但是对于图算法、图挖掘以及机器学习等需要进行多次迭代的计算，MapReduce 模型并不适用。

许多研究已经证明，图处理算法中数据的并行化的缺失是导致 MapReduce 模型不适合迭代计算的根本原因。取而代之，BSP 模型在大图处理领域备受推崇，基于 BSP 模型的图处理系统也有很多，主要有 Google 的 Pregel，开源组织 Apache 的 Hama 和 Giraph。这些模型通常是消息驱动、以顶点为中心的，在单次迭代中会产生大量的消息和网络通信负担。在同一个计算节点内部的消息代价较小，但是跨越计算节点消息传递代价会比较大，所以尽量减少通信边成为优化运行效率的手段。但是在保证图数据的等规模划分又使得跨分区的边的数目最少的数据划分方法在理论上是 NP 难的。在工业界的 Pregel 和 Giraph 默认采用 Hash 的方法将整个图划分，虽然效率很高，也照顾到了各个 partition 的负载均衡，但是有一点很致命的问题就是 Hash 的方法破坏了原先的图的拓扑结构，代价就是巨大的通信开销。尽管如此，很多研究人员还是很推崇这种简单快捷的划分方式。

同时，我们还会考虑另外一个问题，之前我们考虑的问题都是针对一个泛化的迭代图算法，并没有考虑到某个具体的图算法中。那么有没有这种可能，因为不同图算法用到的拓扑性质不同，导致同样的划分方案对某种图算法适用，对另一种图算法并不适用呢？

更进一步，我们能不能提出某种算法，可以根据算法的动态运行，调整划分的策略呢？

三、相关在线划分算法

1、Hash 算法

这个算法其实思想很简单：对于图中的每一个定点，根据设定好的 Hash 函数，计算出一个 Hash，最终将 Hash 相近的点分配到一个 partition。

Hash 函数的计算可以算近乎 O(1)的，而且每个点的计算可以分布式计算，那么我们可以在图的加载阶段就算出每个节点所处的 partition，确实是一个非常快捷的方法，也是各种大图计算平台的默认划分方法。

虽然我们也非常清楚 Hash 算法的弊端在哪里，但是我们却可以从其中发掘到一些非常精妙的思想：我们确实给每个点分配了一个 Hash 值，而且也根据 Hash 值所属的区间去划分的整个图。那么如果我们给每个节点的 Hash 能够反映出整个图的拓扑关系——在同一个 partition 内部的边比较集中，跨 partition 的边较少。如果我们能通过某种方法将 Hash 的算法与图的结构匹配起来，那么就情况不就好很多了吗？

2、locey 算法

根据刚刚对 Hash 分析，我们可以逐步完善这个想法：一开始每个节点的 Hash 是近乎随机的，但是可以通过某种迭代方式使得每个节点的 Hash 值逐渐收敛到稳定的值，而且收敛的值具有某种局部性。

一个直觉的想法是让每个顶点具有某种“质量”，随着质量而来的是因为“质量”而产生的“引力”，顺着“引力”可以让各个顶点之间的 Hash 值流动，随着若干轮的迭代之后，让 Hash 值趋于收敛，然后根据 Hash 的区间划分 partition。

这样一个算法需要我们给每个节点分配一个 weight 权重，而且还需要定义顶点之间 Hash 流动的公式，这两个因素都是这个算法是否有效的前提。

首先给出一个 Hash 流动的公式：

$$Hash_k(v) = Hash_{k-1}(v) + \sum_u \frac{weight[u]}{weight_{neighbors}[v] + weight[v]} (Hash_{k-1}(u) - Hash_{k-1}(v))$$

其中，u 是 v 的邻居节点，$weight_{neighbors}[v]$是 v 的所有邻居节点的权重总和，从公式可以看出，如果某个节点的$weight[v]$比较大，那么他对 Hash 比他小的邻居节点，有很强的“输送”作用；对 Hash 比他大的邻居节点，有很强的“吸收”

作用。在一推一拉的作用下，可以使得相互靠近的节点有比较相近的 Hash 值，这个时候按照 Hash 来划分 partition，可以得到较少的通信边。

但是如何设计 weight 函数呢？也就是如何量化一个节点的“质量”呢？什么样的节点我们会认为是比较重的节点呢？

最简单的想法是认为节点度数比较高的节点是“重”的节点，因为它可以影响更多的节点，在划分的时候也是更多考虑度数多的节点。所以我们可以假设节点的“质量”就是节点的度数，将这个带入 Hash 的迭代计算。当然，我们还可以对度数取不同的次幂，相当于给度数也加上了不同的重要度。

在这里，其实计算出来的 hash 就是 locey 算法的每个节点的 locey 值，这个值一般都可以限定在很小的范围内，这样将所有的 locey 装入内存，方便对所有的点分配 partition。

3、Range 划分算法

刚才的两种算法都是想通过给点进行分类，其实还可以从边的角度来进行思考。现在不妨考虑一个 BFS 的顶点序列，按照访问的顺序，列出所有点。可以预见的是从一个点出发的所有后向点，会分布在访问序列的一段连续的位置，这恰好是局部性的体现。如果我们能够利用这种局部性，是不是意味着这也是一种快速进行划分的方法呢？

现在可以仔细分析一下整个流程：首先我们从一个给定的顶点出发，开始以 BFS 的顺序遍历整个图，可以得到一个访问的顶点序列，然后根据这个序列，我们可以构造出这个顶点序列的邻接表。按照邻接表中的边的熟练，我们可以近乎均匀的划分整个邻接表，得到一个 partition 方案。

这个方案是一个比较粗糙的方案，因为针对网络图数据，大多是满足幂律定律的，也即是有很少的点连接了很多边。那么在 BFS 中，某个顶点的边很有可能就被强行划分到两个 partition，从而导致了这两个 partition 之间的通信边的数目变多。所以还需要优化。

基于定向边的转移可以缓解这个问题。考虑如下一种情况：在某个 partition 中有一个节点 v，它在本 partition 中的边数是 e1，连接到另一个 partition 的边数是 e2，那么如果 v 在原来的 partition 中给所有连接的边发送消息，那么发送的信息量就是所有的跨立边。现在我们考虑把 v 这个节点的备份发送到另一个 partition，每次的更新都会由原来的顶点发送给备份点，这样原先跨立的 e2 条通

信边就变成了块内的边。如果 e2>2，那么这次的迁移就是有价值的。通过这个优化，可以很大程度上缓解通信边跨立的情况。

4、模拟退火算法

首先从模拟退火的场景来说，将材料加热后再经特定速率冷却，目的是增大晶粒的体积，并且减少晶格中的缺陷。材料中的原子原来会停留在使内能有局部最小值的位置，加热使能量变大，原子会离开原来位置，而随机在其他位置中移动。退火冷却时速度较慢，使得原子有较多可能可以找到内能比原先更低的位置。

模拟退火的原理也和金属退火的原理近似：我们将热力学的理论套用到统计学上，将搜寻空间内每一点想像成空气内的分子；分子的能量，就是它本身的动能；而搜寻空间内的每一点，也像空气分子一样带有“能量”，以表示该点对命题的合适程度。算法先以搜寻空间内一个任意点作起始：每一步先选择一个“邻居”，然后再计算从现有位置到达“邻居”的概率。

目前从数学上已经证明，模拟退火算法会依概率收敛到全局最优解。

那么在我们当前的场景中，模拟退火方法应该也是比较适用的。假定现在每个点已经随机属于某个 partition，从退火一开始，每个点就可以选择向自己的邻居移动一步，如果移动的邻居是属于自己同一个 partition 的，那么放弃移动；如果是属于另一个 partition，那么有一定概率移动过去。

在每次所有点都做过一次尝试移动之后，对全局的通信边进行一个计算，如果解更优了，那么无条件接受当前解；如果是变差了，那么根据当前的 Temperature，还有对差距的忍耐程度，有一定概率接受这个较差的解，因为这个较差的解很有可能帮助当前解走出局部最优解的境地。

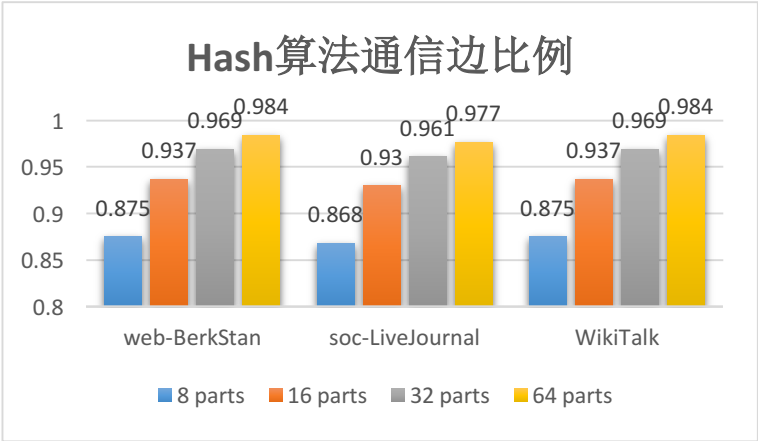
但是，如果是直接放所有的点放开任意游走，迭代 20 次之后，会发现整个划分出现了很大的负载偏斜，边数最多的 partition 会比边数最少的 partition 多十几倍的边，因为这个是滚雪球一样，边数多的 partition 会更容易吸引更多的点。这个时候我使用的排斥操作来一定程度上抑制点从负载少的 partition 中移动到负载多的 partition，具体操作是用当前 partition 的负载，除以目的 partition 的负载得到 k，那么这个点移动到目的 partition 的概率是(1/3)^k，k 越大移动的概率越小。这样能抑制住滚雪球效应。

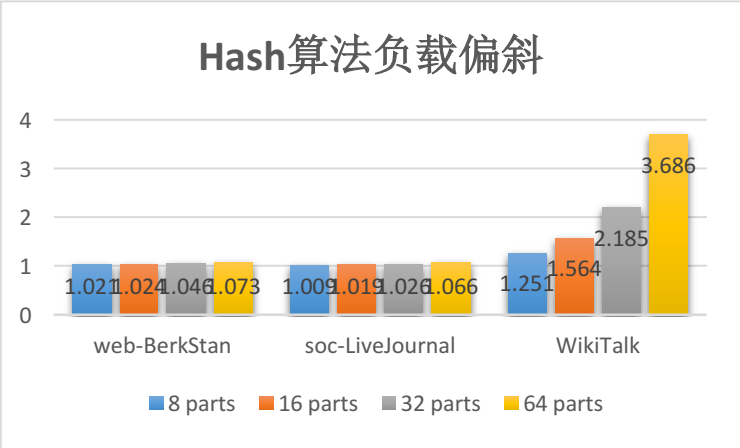
另外，比较好的初始解能够帮助模拟退火算法迭代出更好的解，所以我们希望能够给模拟退火找出一个尽量贪心的解。在实际的操作中我使用的是 locey 迭代 10 次来帮助之后的模拟退火算法。

四、不同图算法对划分效果的检验

之前列举的各种的划分算法，各有利弊，在这里给出全方位的评测。首先来看看对 Hash 算法的评测：

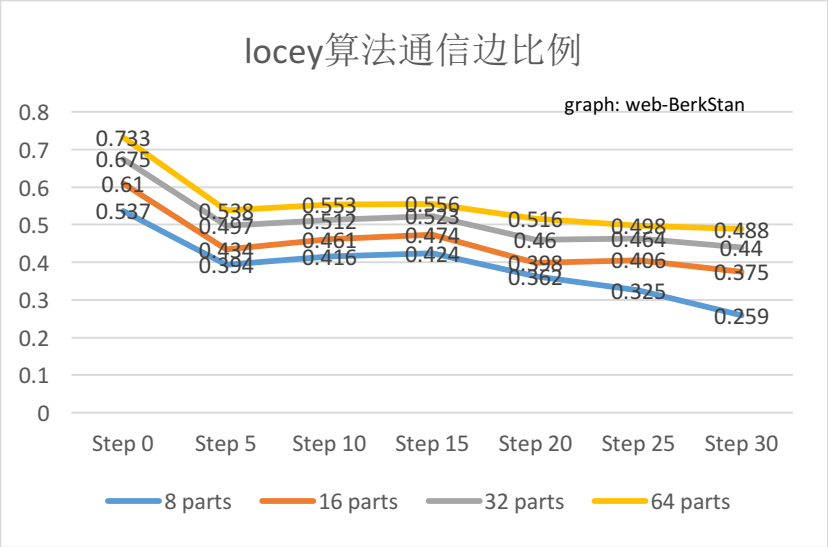
评测使用了三个不同的网络数据，并且对划分的块数也分别测试了。





从上面的两个图片可以看出，Hash 算法的通信代价是非常高的，都在 0.85 以上，并且随着划分的 partition 越多，通信的代价也越来越大。

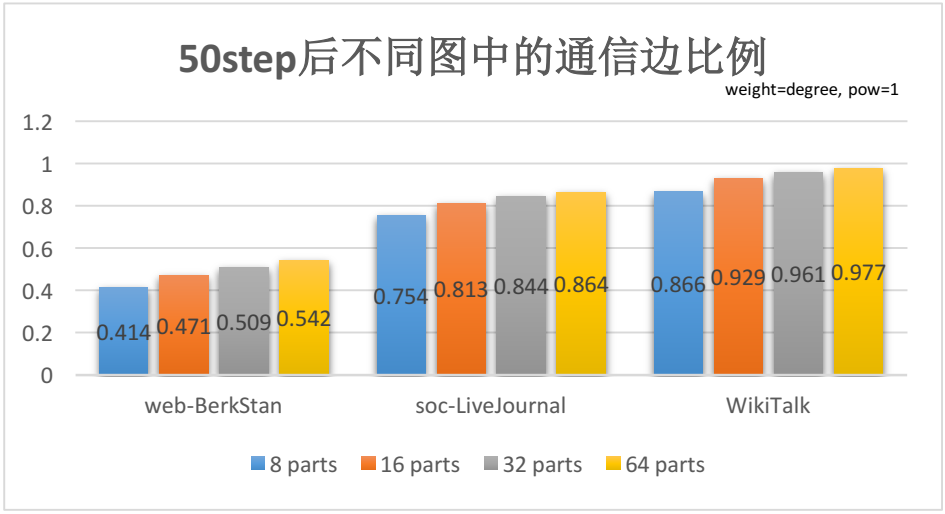
而且从第二个图片还可以看出一个比较特殊的地方，WikiTalk 这个图的 Hash 划分下的负载偏斜竟然达到 3.686 了，说明这个图是幂律定律特别明显的图，有一些极个别的点拥有极大部分的边，这些点分配到任意的 partition，都会引起那个 partition 的负载不均衡。这也会在下面的测试中体现出来。



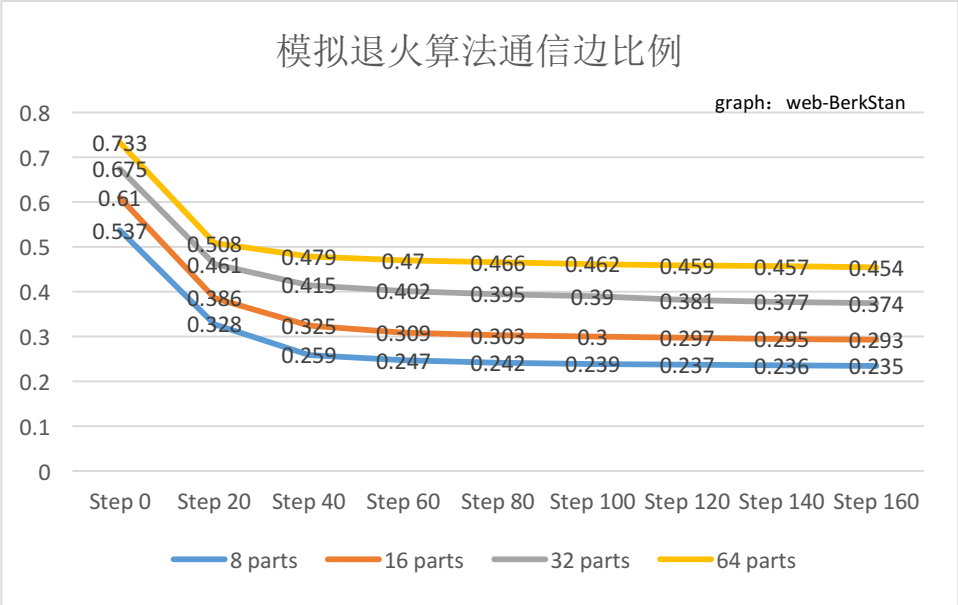
这个图中，我们采用的是 weight 直接取值为度数的结果。可以看出，locey 在前面 20 步骤中的收敛还是不错的，哪怕分成 64 parts 之后，也能在 30 步收敛到 50%以下。

在图中可以看出，在总体下降的过程中会有抖动，也就是在某些时候会有调整之后发现通信量上升的情况。这其实是正常的，当调整带来的好处不能 cover 调整的代价的时候，就会出现这种波动。

下面这个图更加放大了这个代价的作用。在 50 步的调整之后，我们发现其实所有图的通信连都上升了，因为这个时候调整的代价相比较带来的收益太大了，整个图的通信量回到了几乎刚刚开始调整的时候的样子。



以上都是对 locey 算法的一个大致的评测，下面对模拟退火的算法进行评测：



可以看出模拟退火在后期的收敛性还是非常不错的，和之前 locey 算法后期通信量的增加相比，模拟退火在后期的更新是收敛性的，因为温度逐渐冷却下来，可以移动的点越来越少，所以通信量其实是很低的。但是同时也带来一个问题，温度越低，接受较劣解的概率就越低，那么很容易陷入到局部最优解中出不来。我的策略就是每当连续出现 10 次 reject，就强制将当前温度翻倍，目的就是为了增大跳出局部最优解的概率。

加入这个优化的效果是比较明显的，原先的算法到了 100 步之后就开始摇晃不前了，因为这个时候的温度低到不能接受较劣的解了，所以陷入局部最优无法走出。现在这个算法在较低的温度下虽然更新缓慢，但是确实做到了不断找到更优的解。

五、下步工作规划

目前的工作是尝试了不同类型的在线划分算法，并且分析了他们各自的优点和缺陷，但是并不能很好的去综合运用他们。同时，也通过最短路算法和 page rank 算法的运行 trace 发现了其实这两种算法对图划分的要求是不同的，每个 partition 的实时负载对于最短路算法来说是随时变化的，那么我们有没有必要去维护每一个变化？或者说，我们在什么样的情况下需要调整划分？

另外，我们在追随动态划分的时候，动态产生的通信量究竟应该如何均衡？如果我们想要调整到比较好的状态，那么我们势必要了解尽量多的信息，而这些信息在传递的过程中，也一定会随之产生很大程度的通信量；但是如果不能掌握足够信息，就不能进行更好的调整。这确实很一件很难权衡的问题，很多时候，划分算法就在这个时候选择了不再继续操作下去，因为优化划分带来的收益已经

不能抵消划分带来的通信消耗了。但是一种算法进行不下去，是否可以选择另一种效果没有当前算法明显，但是通信消耗也小的算法呢？

到目前的研究很大程度上是建立在各种假设之上，通信量的计算也都是运行的程序自己模拟得到结果，并没有放到真机的环境中进行测试。现在已经在单机环境中搭建出一个伪分布式的 Giraph 框架，并且得到整个项目的源码，可以已经尝试对项目的源码进行修改，计划在接下来的时间中将划分算法嫁接到 Giraph 框架中实现，并且放入真正的分布式环境中进行运行，得到最真实的测试数据。

六、参考资料

[1] Recent Advances in Graph Partitioning, Aydın Buluç, Henning Meyerhenke, Ilya Safro, Peter Sanders, and Christian Schulz。

[2] Bipartite-Oriented Distributed Graph Partitioning for Big Learning，Rong Chen (陈 榕), Member, CCF, ACM, IEEE, Jia-Xin Shi (施佳鑫) Hai-Bo Chen* (陈海波), Senior Member, CCF, Member, ACM, IEEE, and Bin-Yu Zang (臧斌宇), Senior Member, CCF, Member, ACM, IEEE。

[3] PowerLyra: Differentiated Graph Computation and Partitioning on Skewed Graphs，Rong Chen, Jiaxin Shi, Yanzhe Chen, Haibo Chen。

[4] 云计算环境下的大规模图数据处理技术，于戈，谷峪，鲍玉斌，王志刚。

[5] BSP 模型下基于边聚簇的大图划分和迭代技术，冷芳玲，刘金鹏，王志刚，陈昌宁，包玉斌，于戈，邓超。

[6] OnFlyP: 基于定向边交换的分布式在线大图划分算法，王志刚，谷峪，包玉斌，于戈。