大规模图数据中kmax-truss问题的求解和算法优化

一往无前

崔高铭  
 测控技术与仪器&2014  
 齐鲁工业大学  
 中国-济南  
 gaoming.cui@foxmail.com

团队简介

队长：崔高铭

曾就职于华为，齐鲁工业大学2014级测控技术与仪器本科。获得2020年阿里云原生编程挑战赛第二名、2018年华为软件精英挑战赛第四名。

摘要

k-truss是由Jonathan Cohen于2008年提出的另一种稠密子图结构，它由团的概念衍生而来。k-truss放宽了结构约束，仅要求图中每条边至少属于（k-2）个三角形。k-truss是一种分层递进的子图结构， 3-truss ~ kmax-truss刻画了不同稠密程度的网络核心，它们彼此间是包含关系，即一个4-truss可以是一个3-truss的子图，一个5-truss可以是一个4-truss的子图。k-truss在现实生活中有着丰富的应用，例如在社交网络中，k-truss经常用于发掘关系紧密的团体。

任务：输入：无向图G=<V, E>。其中V代表顶点集合，E代表边集合。输出：图G的最大k值，即kmax；以及kmax-truss子图中边的条数。参赛者需要在给定服务器平台和给定数据集上实现求解kmax-truss问题的算法，并对算法进行优化以减少计算时间。

关键词

kmax-truss，truss decomposition，CUDA，GPU

1 赛题分析

k-truss是一个内聚子图，其中每个边都至少是k-2个三角形的一部分。这个子图放宽了clique的概念，可以在多项式时间内计算。k-truss decomposition现有算法包括三个主要步骤：1）确定图中每条边的trussness值，这从根本上说明了每条边是多少个三角形的一部分；2）对于给定的k值，移除trussness值小于k-2的边；3）移除受影响的边，直到图中没有一条边小于k-2。如果图不是空的，那么k的值增加1，并且重复以上三个步骤，直到图为空，我们就得到了kmax-truss。

在只考虑寻找kmax-truss，这时候从k=2开始逐层剥离是非常低效的。在本文中，我们计算出upper\_k和lower\_k，通过k-core将部分边删除，在此基础上进行计算kmax-truss。

2 算法设计

2.1 基本流程

首先采用mmap并行从数据文件中读取数据，然后使用算法1来近似k的上界值。形成k-truss，对于节点数和每个节点的度数是有最低要求的。这意味着对于最大可能的k-truss，需要找到最大的度d，且至少有d+1个节点，d+1的值才能成为k的上界值。在算法1中，我们从最大度到最小度遍历图，并不断聚集节点数，直到达到最大度d，且有d+1个聚集节点数，我们就将d+1作为upper\_k。

|  |
| --- |
| 算法1 |
| int upper\_k = 0; int total\_num = 0; for (int i = n - 1; i >= 0; i--) {  int cur = i + 1;  total\_num += degree\_num[i];  if (total\_num >= cur) {  upper\_k = cur;  break;  } } return upper\_k; |

算出upper\_k后，我们使用upper\_k的1/3作为lower\_k的值。将lower\_k作为计算k-core的k值，删除所有小于k值的边。在删除大量的边的情况下，使用gpu快速计算出残图的kmax-truss，将kmax-1作为lower\_k，缩小搜索范围。

再将lower\_k作为计算k-core的k值，删除所有小于值的边，从支撑度为lower\_k-3开始kmax-truss计算，计算结束后判断是否满足条件，若满足则输出结果，否则将lower\_k减半，重新计算kmax\_truss，直至满足结束条件。

2.2 truss decomposition算法

图采用CSR格式存储，计算边的支撑度时，首先搜索每条边的邻边，第一个大于自身的邻边作为分割点，使邻边列表分割成两部分。如果前半部分的邻节点与前半部分的邻节点相连，则代表三条边形成一个三角形，将这三条边的支撑度加1。最终，得到每条边的支撑度。

从给定的初始k值开始计算，将支撑度小于等于k-2且没有处理的边扫描出来放入cur列表。在剥离阶段，对cur列表中的边进行处理，若存在其他未处理的两条边和与形成三角形，则分为三种情况处理：

1）和都不在cur列表中，则对，都进行更新支撑度操作。

2）不在cur列表中，在cur列表中，且，则对进行更新支撑度操作。

3）不在cur列表中，在cur列表中，且，则对进行更新支撑度操作。

在更新支撑度操作中，若该边的当前支撑度等于k-1，则将该边放入next列表中。若该边的当前支撑度小于等于k，说明有两个边对该边进行了更新支撑度操作，将支撑度加1。cur列表中所有边处理完毕后，如果next列表不为空，将next列表和cur列表交换，继续处理cur列表中的边，直到next列表为空。

2.3 并行化设计

在读取数据文件中，我们将数据文件根据线程数分割为多个bucket，每个线程读取一个bucket，将读取的数据分别存放在线程本地buffer，统计每个线程读取的边数。当所有线程读取完毕，计算每个线程写入的位置，再并行地将线程本地buffer写入到全局buffer。

计算边的支撑度时，因为存储边是双向的，所以会导致同一条边统计两次。我们根据大于u节点的邻节点作为分割点，将邻节点列表分成两部分，在扫描时，当u小于w时，才会进行统计，这样前后没有依赖关系，可以并行化处理。使用gpu并行化时，为了提高线程利用率，邻节点列表循环也使用多线程处理。避免了某些点的邻节点列表过大，线程处理时间过长的情况。

扫描每条边的支撑度，并将符合条件的边加入cur列表中，这个操作本身就可以并行化，但每次加入列表需要原子操作，时间代价较高。所以每个线程设置一个本地buffer，将符合条件的边加入本地buffer中，当buffer满了再加入列表中，同步操作从降低到  ，其中n为buffer的size。

在处理cur列表中的边时，可能存在一个三角形的两条以上的边在cur列表中，如果不进行处理，则会出现重复更新支撑度的操作。所以，我们规定只对最小的边进行处理，这样一个三角形只会处理一次。

3 算法优化

因为支撑度为0的边不属于任何三角形，所以在处理该边时，并不会出现更新其他边的情况，所以可以直接跳过支撑度为0的边。

当图中剩余节点数等于cur列表的节点数时，表示最后所有节点都在cur列表中，这些边不会再更新其他边，所以可以跳过后面的剥离操作。

构建边列表中，查找邻节点时，分不同情况采用线性搜索或二分搜索，在邻节点超过64个时，使用二分搜索，否则使用线性搜索。

4 实验结果与分析

实验机器规格为cpu8核，内存64GB，GPU为1个Tesla v100（显存16GB），cuda11.1。实验结果如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 图 | Kmax | Edge | Time/s |
| com-orkut | 78 | 6859 | 10.415 |
| s23.e15 | 685 | 1327787 | 8.672 |

Table 1: 实验结果

致谢

这次比赛让我深入了解了求解kmax-truss的相关算法及优化，通过阅读大量的前沿论文，以及自我思考，不断突破自己的极限速度。感谢主办方提供这样的平台，让我有幸参与其中，与其他选手共同比拼进步。感谢出题方出了一道这么有趣的题目，还提供给我们环境进行编写及测试。感谢工作人员的辛苦答疑，让我们在比赛中更轻松更快速的解决问题。希望CCF BDCI越办越好。

参考

[1] Conte A , Sensi D D , Grossi R , et al. Truly Scalable K-Truss and Max-Truss Algorithms for Community Detection in Graphs[J]. IEEE Access, 2020, PP(99):1-1.

[2] Almasri M , Anjum O , Pearson C , et al. Update on k-truss Decomposition on GPU[C]// 2019 IEEE High Performance Extreme Computing Conference (HPEC). IEEE, 2019.

[3] Che Y , Lai Z , Sun S , et al. Accelerating truss decomposition on heterogeneous processors[J]. Proceedings of the VLDB Endowment, 2020.

[4] Kabir H , Madduri K . Shared-Memory Graph Truss Decomposition[C]// 2017 IEEE 24th International Conference on High Performance Computing (HiPC). IEEE, 2018.

会议名称：ACM伍德斯托克会议

会议简称：WOODSTOCK'18

会议地点：美国德克萨斯州埃尔帕索

ISBN：978-1-4503-0000-0 / 18/06

年份：2018

时间：6月

版权年份：2018年

版权声明：维护权利

DOI：10.1145 / 1234567890

RRH：F。Surname等。

价格：$ 15.00