SKETCHLEARN 软件实现及其改进

——算法设计与分析 2022 春小班项目报告

兰佳晨 杨仕博 张朝奕

摘要

sketch 是广泛应用于网络测量的一类工具。在之前的 sketch 论文中，针对寻求具体问题统计值的问题，往往采用了将涉及到的参数(譬如 heavy hitter 的阈值)绑定在 sketch 里的手段。通过调研论文，我们认为这种手段存在问题：按上述方法配置 sketch 本质上将数据当成了黑盒，我们并不知道具体数据的分布(譬如，我不知道数据流中具体超过了多少的流在数据流中是 Heavy Hitter)，所以进行 query 的时候需要调节统计参数以适应数据流——这就导致会将统计值调节到一个不适应于 sketch 的可接受的问题，即，某些统计参数的配置会很大幅度降低 sketch 的精度。

基于这个问题，我们从 sketch 框架方面进行了调研(参阅了 Sketchvisor、Sketchlearn、Omnimon 等论文)，最终认为 SketchLearn 可以解决这个问题。SketchLearn 提供了一种从流数据中筛选大流的手段，可以通过统计的方式逐步筛出给定数据流中大小不正常(主要关注过大)的流，以达到不需要借助给定参数筛选“真正的”大流的手段——这就很好地解决了上述问题。我们复现了这个论文，并在复现的过程中基于提供的数据进行了适当的调整和优化。

在此之后，我们认为 SketchLearn 存在着如下两个问题：首先，SketchLearn 对于大流估计的偏差将导致对小流的统计值遭到严重破坏(即，筛选大流时有一部分小流因为统计失误被认为是大流或者自身的统计值被赋予了大流偏差的值)，这导致对较小的大流的估计误差较大(问题 1)；其次，SketchLearn 存在负载不均的问题，流键的特定分布会导致某些统计信息失真明显，不但需要冗余的 Sketch，还会最终影响统计，这在短时间的统计内将会产生较大的误差(问题 2)。

针对第一个问题，我们认为过滤掉会影响小流的过大流会提高数据的精度，以此实现了适合于 SketchLearn 的大小流分流策略——通过统计两种误差值，我们发现我们方法的误差值降到了原始误差的 57%甚至更少。针对第二个问题，我们认为筛查出来分布过于偏激的 bit 位对应的 sketch 是必要的，我们使用类似陶片放逐的策略指出了分布偏激的部分并以此信息调优了统计方法，在模拟短时间的统计中得到了很好的效果。

我们的代码和 SketchLearn 笔记已经开源：可参考 https://github.com/CarambolaCup/SketchLearnSoftware，其中 sketchlearn_debug 是原始的 sketchlearn，version1 和 version2 是我们分别进行了上述两种改进之后的代码，使用的数据是 formatted00.dat

SketchLearn 简介

(详细的介绍我们放在了 https://github.com/CarambolaCup/SketchLearnSoftware/blob/main/Sketchlearn%E7%AC%94%E8%AE%B0/Sketchlearn%E7%AC%94%E8%AE%B0.md，这里只提供简单的介绍性文字)

SketchLearn 的想法是，针对输入的流键，除了一个统计全局的 CMsketch 外，我们使用多个 sketch，从每一位的维度维护流频率的统计信息。使用这些统计信息，我们就可以推断出某一个 sketch 的桶中，哈希进超过某个比例的阈值的大流的可能值、这个大流对应的这一位是 0 还是 1、这个流大小的估值，甚至这一位估计错误的概率等等的信息，使用这些信息我们就可以学习到数据流中真正大流呈现的分布，以及大流估计的具体值，以此便可以准确回馈多种多样的 query。下面详细说明这些内容。

首先，我们使用(l+1)个 CMSketch，其中 l 是流键最长可能长度，其中第 0 个 sketch 统计所有流的信息，第 i 个 sketch 统计所有第 i 位为 1 的流的信息(i=1,2,...,l)。第 k 个 sketch 的第 i 行 j 列我们记作 V[k][i][j]，当 k 跑遍 0 到 l 时所有涉及的桶我们记为栈(i,j)。

注意到，如果给定流第 i 位为 1 的概率为 p[i]，那么在所有流频率相近的情况下，输入每个流时第 i 个 sketch 哈希到的流量(1 或 0)服从 0-1 分布，其和服从二项分布，当流量足够大的时候，此和可以近似视为正态分布，于是对任意 k,i,j，V[k][i][j]/V[0][i][j]服从正态分布。并且，由于每个 i,j 可以视为第 k 个 bit 的一次采样，通过整个 sketch 中的数据我们可以推断出这个 sketch 对应的正态分布的均值和方差，从而确定对应的正态分布。

于是，大流的到来必然将会产生过分偏离正态分布的数据。如果我们想找出栈(i,j)里超过 $\theta \times V[0][i][j]$的流，我们可以通过每个 bit 的统计信息(是否这个 bit 统计到/没统计到的值超过这个阈值(对应这一位应该是 0 还是 1)，或者假定这一位有超过这个大小的流，计算出来这一位可能哈希到这样一个流的概率和对应这一位是 0 或 1 的概率)得到这一位有超过这个阈值的流量的流、并且这个流这一位是 1(或 0)的概率。依靠这些概率，我们可以得到一个描述这个可能大流的正则表达式(上述概率不接近 0 或 1 的时候认为这一位是'*')，之后通过枚举每个可能满足上述正则表达式的流，通过哈希函数和 CMSketch 真正统计到的数据判定这是不是真的大流。另外，通过“筛掉多少流可以让这个 sketch 更符合正态分布”这一点，我们可以给出对应流的估值(这个数据可以根据 CMsketch 得到进一步的优化)。

注意到，为筛出所有可能大流，我们只需要将 θ 不断下调即可。但是，什么时候终止呢？答案是，当剩下的流量与总流量的比值很好地符合了正态分布即可停止了，验证这一点只需要依据 $3-\sigma$ 法则设定偏离的比例，当且仅当所有 sketch 里的数据的偏离量都落到了给定的范围内的时候，我们就结束筛选。

使用上面的方法，我们很好地筛查出了所有可能的大流，并且没有依赖于输入的阈值。

在本版本的实现中，在配置上，我们做了一下几点小的我们认为可能可以优化性能的优化：

第一，如果正则表达式中的''的数目超过了一定阈值(此处是 11)，我们认为实际上这个栈不应该有符合设定的大流，停止搜查——这既不会影响效率，也可以大幅减少搜寻符合正则表达式的流的时间，因为''过多说明存在大流的概率太小了。

第二，如果我们通过统计值估计出的流量太大(超过最小 sketch 中统计到的流量的若干倍(此处是 2 倍)，并且非常接近第 0 个 sketch 中的值(此处是超过 95%的 V0 中的值))，那么我们认为这个流是假的流，即不在原始流量里但是被正则表达式包含的流。真正的流估计值不应该如此极端，所以这一步是合法的，并且这样可以省略掉因为筛选可能流耗费的时间，减少假阳性(引起的多删流)引起的误差。

第三，我们在终止条件中限制条件的界根据 $\log \theta$ 的值线性地放宽，这样可以在数据真的已经被竭尽全力地筛查之后有效地停止算法，也充分考虑到了服从正态分布这个事情的偶然性。

问题分析

我们来具体阐述一下我们发现的两个问题：

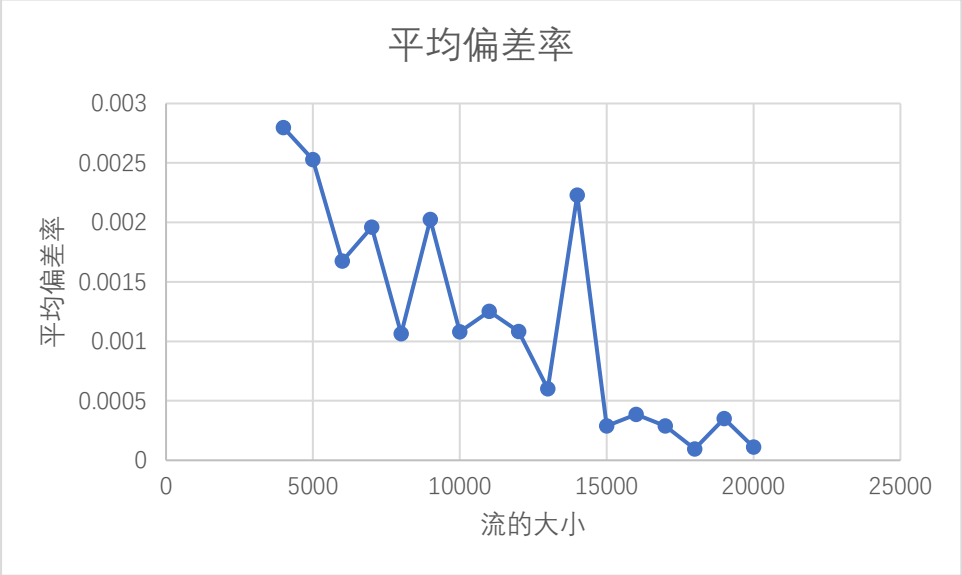
问题 1：我们观测到，对于给定的数据，越大的流精度越高。而对于较大的流(足够大到应该纳入考虑的范围内，但是不是最大的流)sketchlearn 测量得到的精度相对要差很多，而这会导致 sketchlearn 的精度不够。在给定的数据集中，使用 3*9000*流键长度的 sketch(此时空间足够大，性能也相对好)平均误差大小(测量数据与真实数据的比与 1 的差距的平方)与流的出现的频率关系如下图所示。

分析原因，我们认为出现这个问题的原因是通过统计公式推断小流频率所用的数据是筛掉大流之后剩余的数据，而这部分数据不可避免地夹杂着筛大流过程中存在的误差。譬如，一个桶中哈希到了一个大小为 10000 的流和一个大小为 1000 的流，我们对大流的精度高达 99%，测量数据是 9900，那么

sketchlearn 会认为小流的统计数据为(1000+(10000-9900))，它的精度将在90%上下，这是很不好的。

换句话说，小流的误差将会在逐级筛去大流的过程中逐级叠加，导致误差过大。

问题 2：我们观测到，sketchlearn 具有负载不均的问题。具体来说，出现了统计某些位的 sketch 计算的击中概率很接近 1 或 0 的问题，即输入的流中这几位只有单一的 0 或 1。这些偏激的统计值不仅会引入负载不均的问题，还会导致统计值出现偏激的取值，从而影响计算具体流频率的结果。我们的实验表明，对于小数据而言，偏激的取值将会导致估算结果出现约 10%的多余的偏差。



实验配置

使用的数据是学姐在小班群中提供的两组数据，我们提供的代码提供了在两组数据上进行测试的接口，一共有 27553679 个流参与测试。

我们使用的 sketch 大小为 3*9000*l，其中 3 是行数，9000 是列数，l 是流键的 bit 数目，对于裁剪前后的流分别为 64 和 104。这个数据是基于原论文使用的 sketch 内存大小开辟的，但是考虑到原论文可以在网络间合作，而我们考

察单个路由器上的结果，因此我们在行数上在原论文的基础上增加了 1 行，以此更好地展现其性能。

对于问题 1，我们希望加入陶片放逐的方法减轻问题，我们将展示我们新需要的空间与精度的关系以表现我们方法的优越性。对于问题 2，我们希望屏蔽掉负载过度不均衡的 sketch(在软件部分可以 free 掉这个 sketch 以减少内存使用)，我们将展示在小数据上屏蔽掉这些 sketch 之后的精度上升程度。

陶片放逐法

由于大流与小流存在数量级上的差异，SketchLearn 对于大流估计的微小偏差将导致对小流的统计值估计与实际值偏差很大，为此，我们对大流的估计必须更加精确。对此，我们的想法是，在读入流的同时，通过陶片放逐法将流数目显著大的流先提取出来，通过控制参数，我们可以保证将大于一定比例的流提取出来，并估计这些流的大小，实验表明，这些流数目的估计的精度是极高的，这使得我们对大流的估计的偏差显著减小，从而减小了后续对提取较小流的影响，此外，由于多级 sketch 中已经不存在显著大的流，提取过程也可以加快。下面详细说明我们的实现方案：

首先，我们有 h 个桶和将流均匀映射到这 h 个桶的哈希函数 heavy_hash。每个桶记录流的信息：流 ID、正投票数(vote+)、负投票数(vote-)和标志(flag)。vote+记录属于此流的流大小；vote-记录其他流的数量；标志表明此流是否可能曾经被驱除进多层 sketch。对于每一个读入的流 f，我们尝试将其插入到流键哈希对应的桶内，如果桶是空的，则插入（f，1，0，false），其中 false 表示桶中没有发生驱逐，如果桶中流与 f 相同，将 vote+增加 1，如果不同，vote-加 1，并且判断$\frac{\text{vote}-}{\text{vote}+}$比例，如果其小于 η，则将流 f 输入到多层 sketch 中，如果不同，则将桶中流逐出到多层 sketch 中，并将桶更新为（f，1，1，true）,true 表明多层 sketch 中也可能有 f。

之后，我们将大流集合初始化为这 h 个桶中的流，对于标志为 false 的桶，vote+即为流的准确数量，对于标志为 true 的桶，vote+记为流数目，也可以做到非常精确。在最坏的情况下，大流哈希冲突，由于多级 sketch 采用了多个与

heavy_hash 不同的哈希函数，我们仍可以做到对其中一个流很高精度的估计，另一个大流则可以在后续 sketchlearn 中提取出来。

使用以上方法，我们提高了对大流中显著大的一部分流的估计精度，从而明显减小了大流估计误差对小流的估计值的影响。

评估指标：

对于一个流 f，其真实出现的次数为n_{f_1}，算法估计值为n_{f_2}，我们统计n_{f_1}或n_{f_2}大于等于$\text{threshold} = 4000$的流F，和F中小于$\text{threshold}' = 9000$的流F'，指标定义如下：

偏差率（Deviation rate，DR）：

$$\sum_{F}\left(1-\frac{n_{f_1}}{n_{f_2}}\right)^2$$

加权偏差率（Weighted deviation rate，WDR）：

$$\sum_{F} n_{f_1}\left(1-\frac{n_{f_1}}{n_{f_2}}\right)^2$$

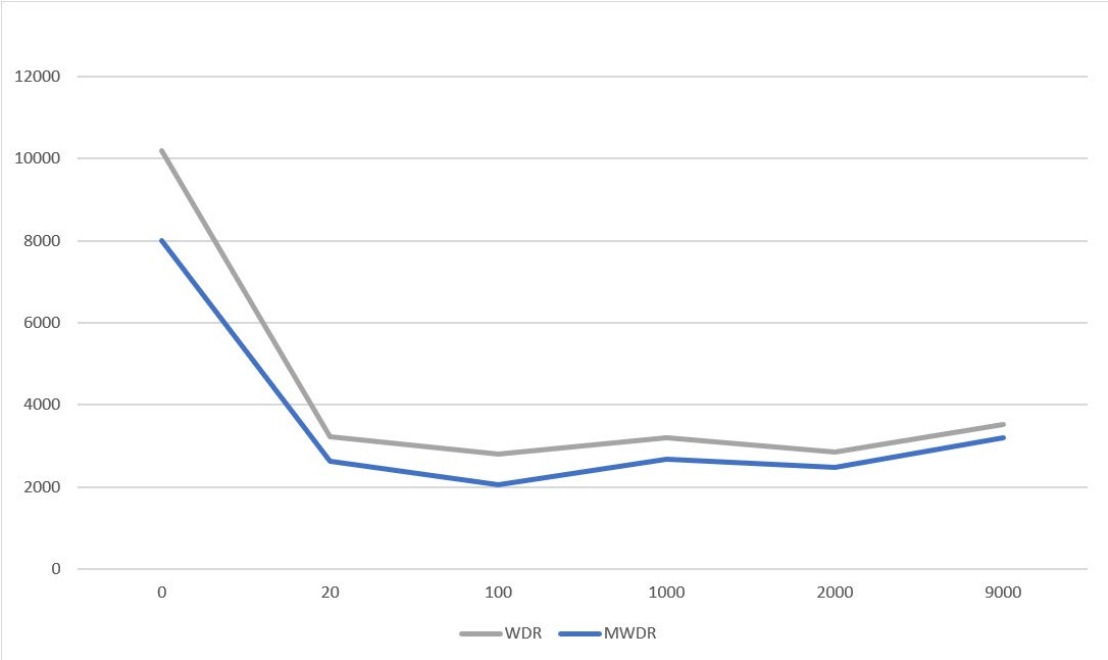
较小流偏差率（Minor flow deviation rate,MDR）：

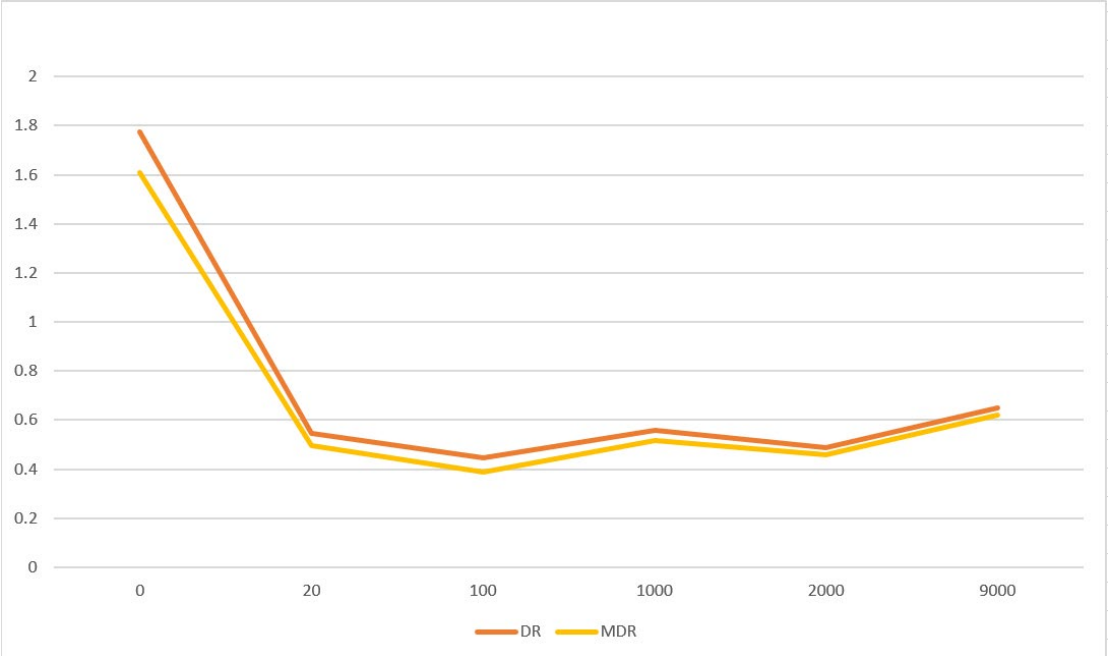
$$\sum_{F'}\left(1-\frac{n_{f_1}}{n_{f_2}}\right)^2$$

较小流加权偏差率（Minor flow weighted deviation rate,MWDR）：

$$\sum_{F'} n_{f_1}\left(1-\frac{n_{f_1}}{n_{f_2}}\right)^2$$

实验配置同上,分别读入十个小数据,并在不同 heavy_hash_length 下运行。在实验中，我们观察到，加入陶片放逐后，各项数据都能优化，且可稳定在原版本的$\frac{1}{4}$左右。陶片放逐所占用的空间比原方法有数量级上的差距，但可以大幅度优化算法运行性能。





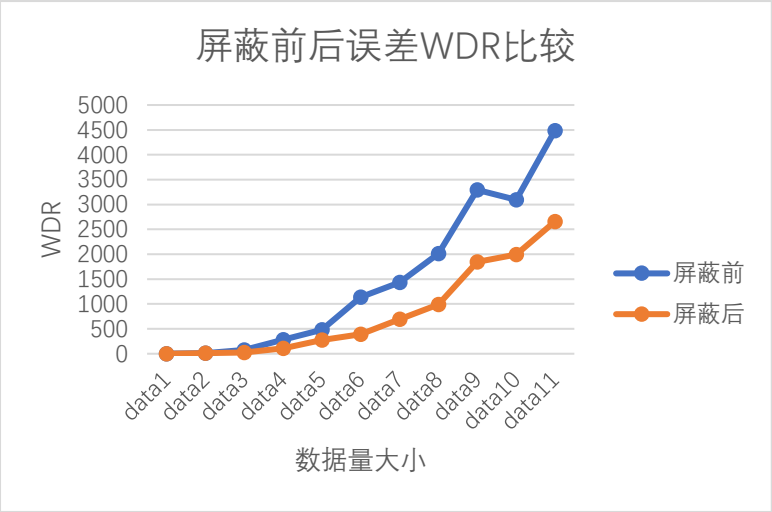
屏蔽不均衡 sketch

在较小的时间窗口内，流可能具有较为明显的特征，具体体现在几乎所有的流的某一位都是 0 或 1。如果在几乎所有流第 k 位均为 0，则这个 sketch 接近于一个零矩阵，而如果在几乎所有流第 k 位均为 1，则 sketch 接近于全局的 CMsketch，这是极不均衡的。

对此，我们维护了对于 1 个 bit 的计数器，并在判断流的第 k 位是否应计数时，也相应地修改 0 计数器与 1 计数器。读取流完毕后，将比值超过阈值的 sketch

释放掉（我们实现的代码中并没有真正的释放，只是弃用此 sketch）。

然后我们对不同数据量的网络流进行了实验，结果如图：



发现平局误差减小了 1/3 左右。我们推测屏蔽 sketch 对效果的优化的原因是，对于零矩阵，SketchLearn 算法的计算过程会导致较大的误差，当不将这些极端值纳入计算时，算法对较大流的估计会更准确，因此屏蔽 sketch 对性能也有一定的优化。

总结

在本项目中，我们复现了 SketchLearn 这篇论文以解决参数配置影响 sketch 精度的问题，在过程中发现了 SketchLearn 的大流会对后面较小流的提取产生干扰，以及 sketch 不均衡的问题，并给出了相应的优化方案，即利用陶片放逐法对大流进行初步统计和屏蔽极端 sketch，并且对准确度有了良好的优化。