A Sketch-Based Distance Oracle for Web-Scale Graphs

2015年10月3日

19:25

* + **摘要**

我们的研究是关于在大图中，例如网络图、社交网络等，计算任意两个节点间距离的基本问题。我们的目标是实时回答对任意两个节点的距离查询。由于标准的最短路径算法的计算代价较高，因此我们的方法是把耗时的求最短路径移到线下计算，在查询时，只需查找这些预处理的值然后做简单快速的计算即可。更具体的说，在线下阶段，我们为图中每个节点计算并保存了少量的“sketch”，这样在查询期间，我们用源和目的节点对应的“sketch”来估计他们之间的距离。

* + **介绍**

大图已经成为一个普遍的工具来表示真实世界中的数据，例如，我们如今通常感兴趣的搜索引擎和利用网络图的社交网络站点以及其后台等。其中大量的请求、查询交互在实时地发生着。一个对高代价操作的处理方法是将其放到线下做预处理并保存相关数据，从而当需要实时操作时能够快速的做出响应。例如，PageRank（网页排名）就是由线下对网络图做一个高代价的特征向量计算然后线上为每个结果做简单的PageRank等级查看。

在大图中一个基本的操作是找出任意两个节点间的最短路径。这个问题不仅是许多算法常见的基本组成部分，而且其本身也是一个很有意义操作。例如，在一个社交网络中，一个人可能会感兴趣去寻找连接到名流的最短的朋友序列。在网络图中，连接两个URL之间的短序列可以代表两个页面一定程度的相关性【13,20】。然而，这些大尺寸图中，最短路径的计算是具有挑战性的。在一个包含成千上亿节点的网络图中，运行Dijkstra众所周知的最短路径算法会花费几个小时，甚至几天。另外把一个web规模的网路图保存在单个机器的主存中是不可行。甚至在分布式环境中，如果计算是并行的，由于Dijkstra计算的顺序依赖性也会需要大量的通信。此外，在一次实时计算中，对于单次的最短路径查询只会占用少量的资源（内存访问，CPU周期）。当我们想要去做到这一点以最少的实时延迟，那么每次距离查询必须使用少量的资源。在本文中，我们研究了大图中实时估计两个节点之间距离的问题。

一个简单直接的方法是做一次线下计算，通过dijkstra算法计算任意两个节点的最短距离并保存在磁盘上。在这种背景下，响应一次线上最短路径查询仅仅需要磁盘查找；空间复杂度是图的节点数的二次方。对于一个包含成百上亿节点的网络图来说这种方法根本是不可行的。为了减小空间复杂度，我们的算法保存关于每个节点的少量辅助信息来提高在线实时计算的速度。这些辅助信息随后在每次请求或者查询时被来在线上计算。我们可以认为这些被存储的辅助信息就是每个节点邻域结构的草图（sketch）。由于“sketch”的计算是离线计算，所以一次性预处理花费再多的资源也是可以承受的。

对一个大的对象集合线下计算sketchs来辅助线上计算的方法已经广泛的用于许多应用中。例如，一些搜索引擎计算文档的小sketch来检测在查询时重复的文件; 见[5，10，4]和其中的参考文献。这避免了去比较大的文档，这是耗时的。取而代之，通过比较这些文件小的sketchs来判断文件的相似性。

对距离的特定目的计算sketch也被称为距离标记。一些论文研究了每个节点允许距离计算的标签的大小的问题，包括Gavoille等人。 [14]，Katz等人。 [16]，和Cohen等人。 [7]。

标量嵌入的领域涉及映射一个从高维空间到一个低维空间的点集，使得该失真最小化。如果每个点可以投影到一小维数的向量作为sketch，这样使得距离大约保存。Bourgain [3]研究的经典结果就展示了这样的嵌入如何对于某些距离度量实现。

在估算距离的另一项工作线是spanner construction的研究。一个spanner是给定图的稀疏子图，使得对于任意两个节点，在这个稀疏图的距离大约为实际距离。虽然spanner占用空间小，但是它们并不能完全提供每个节点的sketch; 因此，用于估计距离的在线计算可能需要很长的时间。一些理论上的关于spanner的高效算法被提出，例如Feigenbaum等人 [12]，和Baswana[2]。其他基本结果在此研究领域的有Bartal[1]和Fakcharoenphol等人。 [11]。

Cohen等人[6]提出了一个估计距离的方案，在一个有向图（无向图）中使用2-hop算法覆盖所有路径。然而，在一个大图中找到一个给定路径集合的最优2-hop覆盖的代价是昂贵的。另外，如此一次覆盖的大小是使得他们的方案在一个大图中很难去实现。在一些研究中，例如Goldberg等人[15.9]，都集中在回答确切的公路网络中最短路径的查询。这些算法在查询的时候充分利用了一小部分预先计算的连接源/目的地的路标和捷径。路标和捷径都是被十分谨慎地挑选出来的，且使用的算法也是专门针对道路网络的结构。我们的算法可以被看作是利用随机采取的路标集合;因此，它甚至可以被理解为一个简单的算法，它充分利用随机抽样得到的“地标”能够很好得在其他复杂图中运行，如网络图。

* + **我们的贡献**

在这项工作中，我们设计算法来计算每个节点的SKETCH，并证明怎么用SKETCH来实时估计任意两个节点间的距离。虽然在计算距离sketchs中没有多少经验研究，但还是有许多对于embeddings[3,1，11]和spanner[12，19，2] 等前述理论研究的算法文献。所有这些算法大概只工作在无向图中，有些是复杂的而且很可能是不切实际的。Bourgain[3]的经典结果展示了怎样把一个图投影到一个低维空间中，给出一个小的sketch估计距离的复杂度为的因子。Matousek[17]之后展示了同样的算法，其能够逼近2c-1因子用sketchs尺寸的。但是，我们发现在我们的实验中Bourgain的算法在估计距离时表现非常糟糕。另一方面，我们提出了一个算法对Thorup和Zwick[19]的算法做了必要的简化并提供相同的理论保证：我们的算法能够逼近2c-1因子用sketchs尺寸的（注意，当 c = log n ，是的因子的逼近用sketchs尺寸的）。此外，我们的算法实现起来更简单。而这两种算法不同尺寸的采样种子集和找到在每一个集合中最近的种子，Thorup和Zwick的算法需要在一个节点的sketch之外存储额外的数据，也就是所有那些比在下一个种子集合中最近种子更接近的按照尺寸递减排序的节点ID。这增加了更多的复杂性在线下计算过程中，也导致了一些超出了我们在线上步骤的额外的检查。在本文中，我们证明，sketch之外的附加数据是不必要的，简单地保持种子ID及其相关距离就已经足够了。我们发现在我们的实验中，即使在一个大的网络图中，不管是有向还是无向的距离，我们的算法依然能够给出很好的估计——这个距离误差通常能够精确到2到3的范围内。我们的算法对于有向距离表现良好的事实是令人惊讶的，因为目前对于有向距离还没有已知的sketch算法; 事实上，众所周知，这是不可能在最坏的情况下对于任意的有向图，这也表明网络图具有一些特殊的性质。对于未来的研究，理论证明和实验观察上的差异也是一个有趣的研究领域。然后，对图中的每个节点上，发现在这些种子集的最接近种子。

我们的算法背后的基本思想如下：在线下计算中，从图中采样少量的节点集合（种子节点集合）。然后，对于图中的每个节点，找出距离这些种子集最近的种子。对于某个节点的sketch其简单地由最近的种子和到这些种子的距离组成。然后，在线上计算时，可以使用到这些最近种子的距离来估计给定节点对之间的距离。一种方法是去检查两个节点的sketch之间是否有公共节点。给定一对节点u和v可以通过查找其sketch中共同的种子来估计它们之间的距离。如果是节点u和节点v的sketch中共同的种子，那么两个节点的距离可以通过计算到公共种子距离的和来估计。我们也注意到，我们的算法对于实际距离产生了一个上限; 而Bourgain的算法产生下界。

实验中我们观察到我们的算法对于有向图或者无向图都有很好的表现。我们的实验是在一个有2002次抓取的网络图上进行的。我们把估计距离与通过Dijkstra算法得到的精确距离进行比较。这个实际的无向距离是在1到15的范围内，有向距离（被一个有向路径连接）是在1到100范围内。我们用于检查的性能样品从每个这些距离的对。我们从每个这些距离对中采样用于检查性能。对于实际距离为d(u, v)的节点对u, v我们计算每一个我们算法所获得估计距离的中位数。对于无向和有向距离，我们都独立的做了这些工作。

* 1. **我们的算法**

我们接下来将将描述我们的算法如何计算每一个节点的sketch，这些sketch可用于对于任意两个节点u和v执行线上距离计算。为简单起见，我们只在本节做无向图的工作，然后在第4节中推广到有向图中。

我们用*G*表示一个图，*V*表示其所用节点，*E*表示图所用边，*n*表示图的节点个数。节点u到节点v之间的距离用表示。我们的目标是通过预处理过程，以这样一种方式存储图，给定任意一对节点u和v，我们能够实时地用少量计算来估计距离。有一点很重要，网络规模的图通常包含数百亿的节点和上亿的边。预计算由计算每个节点u的SKETCH[]组成。而实时计算任意节点u和v之间的距离应该只需要读取和。

我们会用表示通过sketch算法获得的对的估计。本文中我们用到的另一个记号是 ，其表示节点到一个节点集合的最短距离。

**Offline-Sketch算法**

如前所述，在我们的算法的基本想法是去采样一组种子节点，然后存储每个节点距离这些集合最近的种子及其距离。这能够被高效的实现从这个种子集合仅通过广度优先搜索（BFS）算法（注意，甚至可以找到最近的节点，通过在执行BFS时设定适当的节点ID）。给定一个节点u和一个种子集合*S*，其中是距离u最近的，则*s* 就是节点u在集合*S*中的种子。对于尺寸为 的集合，去找到最近的种子及其距离的复杂度是二次方的，参见函数 (见算法1)。函数（见算法2）会重复 *k*次，每次用不同的随机产生的集合，最终返回的所有采样的合集。所有线下操作执行完后，会为图中每一个节点保存一个。

为了在估计源节点u到目的节点v之间的距离，种子节点必须位于节点u到节点v的路径上。要在这个有向路径上，一个节点的入度与出度必须是非零的。因此，我们应该只需要考虑符合这样要求的节点作为种子集合的候选者。在无向图的情况下，我们只需考虑度数至少为2的节点。

**算法1** ： OFFLINE-SAMPLE(*G*)

**输入：**一个无向图*G*和其中一个节点u

**输出：**SAMPLE[u]，一个关于节点u的临近种子节点集合

1：另 ，然后从所用结点中均匀的 采样r+1个样本的大小分别为1,2,, ... , （其中忽略度数小于2的结点）。这样我们得到一个样本集合，，.…，

2 : 对每个结点u分别计算到各个样本 的（，），表示 中距离结点u最近的结点（只需要一次BFS算法就可以得到结果），。最后得到SAMPLE[u]={(，),.…,(，)}。

**算法2 ：** OFFLINE-SKETCH(*G*)

**输入：**一个无向图*G*和其中一个节点u

**输出：**结点u的SKETCH[u]

1 ： 运行OFFLINE-SAMPLE(G) k次每次独立随机采样。

2 ： SKETCH[u]就等于k次运行返回的SAMPLE[u]的并集。

以上规则适用于任何图。另外有一些其他规则可能进一步提高我们的算法的准确度（例如，让采样过程偏向高度数节点），但是它们依赖于图的拓扑结构（例如，“哑铃型”图中，偏向高度数节点是一种坏的策略）;因此，在本文中我们没有考虑这些情况。

**Online-Common-Seed算法**

我们现在我们主要的算法使用sketchs，估计距离。对于给定的节点u和v，算法Online-Common-Seed（见算法3）通过观察从节点u和v到任何同时出现在和 的节点w的距离。从节点u经过w到v的最短路径的长度是；注意和都是包含在sketchs中的不需要去计算。我们取所有这样的公共种子w的最小的来作为估计的距离。这是很容易表明，通过Online-Common-Seed估计的距离是实际距离的上界。

*观察3.1 由得到的估计距离 是实际距离的上限。*

证明。该*算法*考虑了节点u和v的两个sketch的交集中各种节点*w*。这些sketch包含了每个节点*w*分别到节点*u*和*v*的准确距离和。根据三角不等式，。注意我们取关于这些*w* 的和的最小值，而不是这些节点的每一个，三角不等式成立。观察如下。

*定理3.2 由*k（k=）*次*ONLINE-COMMON-SEED(u,v)*得到的估计距离其接近实际距离的可能性为*2c−1*；*d(u, v) ≤ ≤(2c − 1)d(u, v)。

**算法3 ：**

**输入：**图G的两个结点u和v

**输出：**的估计值

1： 获取SKETCH[u]和SKETCH[v]

2： 找出SKETCH[u]和SKETCH[v]的公共结点的集合。注意，假设图G是联通的，那么至少会有一个公共的结点，即执行一次BFS最少会有一个尺寸为1的集合。

3 ： 对每一个公共结点计算d(u,)和d(v)。

4 ： 返回最小的d(u, *w*)+d(*w*, v)。如果没用公共的种子结点就返回∞。

证明。让 ，、分别表示结点u、v在rd的半径范围内的结点集合；也就是说，距离节点u和v的rd距离范围内的所有节点。

考虑在 或 中的节点。如果一个种子集合其有一个种子*w*刚好既在合集中也在交集中，另外*w*是节点u和v的sketch中公共的种子。这是因为它是到节点u和v都最近的种子。我们假设这样的种子集存在，这会导致sketch的公共种子距离节点u和v的距离最多为。为简单起见，让我们考虑当的情况。恰恰，我们观察到如果 至少是一个常量（比如），然后当我们得到种子的概率为，就会有一个常量的机会刚好有一个种子既在合集中也在交集中。如果种子被采样的概率为，那么该事件发生的概率至少为，恰好有一个种子的概率为且这个种子有的概率在交集中。由于我们正在尝试种子集合的尺寸是2的不同次方，这个概率也会接近2的次方。这样，如果对于1 .. log n范围内的任意i，这里有一个常量的概率去找到一个公共的种子在节点u和v的距离范围内。否则，意味着对于所有的有。

但是注意到，左边的集合包含距离节点u或者v最多rd的节点，另外，所有这些节点的最多是在节点u或者v的的范围内，那也暗示了它们都存在于。所以如果对于所有的有，这意味着，同时可以得到也是可能的。如此，肯定有一个r的值使得。由于我们尝试每个尺寸k次，对于常量k，我们可以忽略不计失败的概率。对于任意的*c*同样的证明可以给出；我们展示为i，，得到；如果我们重复 次，我们能够以一个高的成功概率得到一个的近似。

**算法4 ：**

**输入：**图G的两个结点u和v

**输出：**的估计值

1： 获取SKETCH[u]和SKETCH[v]

2： 对于每一个种子集合*S*，从SKETCH[u]中获取，从SKETCH[v]中获取，计算。

3 ：遍历所有的种子集合*S*返回最大的。

通过查看集合和——注意对于任意在交集中的点其到节点u和v的距离总和最多是。

**Online-Bourgain算法**

我们把我们的算法性能与Bourgain的众所周知地将一些标量空间嵌入到低维空间中的那个算法做了比较。我们描述了这个算法在ONLINE-BOURGAIN（见算法4）。我们注意到，同样的sketch之前也被用。然而，代替找出在两个sketch的交集中的节点，我们发现从节点v到所有的种子集和从节点u到所有种子集的距离是相似的。这个——这两个向量差的范数给出了节点u和v之间实际距离的下限。

此外，Bourgain也证明了如果我们设定k为（这里k是被采样的尺寸的集合的数量），这个距离可以精确到的常量因子。通过使用2的次方尺寸的种子集以及每个尺寸使用k次，可以理论确保估算值的精确度。我们做出了轻微的调整，可以扩展到有向图。当延伸到有向图时，可惜就不能证明一个的近似度。然而，我们做了简单的观察了发现距离估计是实际距离的下限。请注意，在算法中，对于无向图，我们考虑了的绝对值。

*观察3.3 由得到的估计距离 是实际距离的下限。*

证明。该证明是从三角不等式推演。对于任意节点集合*S*我们知道。所以（由对称性得）也能被证明。看到这一点，让），其中是*S*中最近的节点。由三角不等式，可以得到。由于是节点u在*S*中最近的节点所以。现在估计值能够在线下阶段对于所有集合*S*计算的最大值来获得。但是对于每个*S，*这个差是的界限，最大值也是的界限。

Matousek[17]证明了Bourgain的算法给出一个近似，用了尺寸的sketch。然而，注意这个结果只是针对于无向图。

*定理3.4 当*k=*由计算得到的估计距离有一个高的的可能性 因子的近似；。*

证明。在[3]中给出。

* 1. **推广到无向图**

在上一节中介绍的算法可以容易地扩展到有向图的情况下。但是，也有除了我们已经展示的上限和下限，其他的没有已知的理论保证。相比通过重写改变的子程序的无向图算法，我们阐述在有向图中算法的主要区别。

**对Online-Common-Seed的修改**

如前所述，上界的精确度可以通过线下计算阶段提高采样集合的数目来改善。重要的是要选择不同的尺寸的集合（以获取合适距离），以及每个尺寸多组集合（以减少恰好找到从节点u或者v同样的最近节点的敏感性）。

在有向图的OFFLINE-SAMPLE版本中，我们为每个节点u计算两个sketch，一个抓取到最近种子的距离，一个抓取从最近的种子到节点u的距离。对于每一个被采样的集合*S*，我们找出*S*中的使得（即是最小的），找出*S*中的使得（即得是最小的）。对于所有的节点u这可以通过两个从*S*开始的BFS高效的得到。一个从S开始迭代的去遍历输入边。另一个用这个输出边。

线上距离计算算法考虑了节点u和v的sketch中所有的w。但是，计算有向距离，我们考虑所有sketch中的w对应于从节点u到*S*的距离，对于节点v，sketch对应于从*S*到v的距离。这个距离估计被计算像之前一样，。

在有向的情况中上限的证明同无向的情况类似，应用有向距离的三角不等式。

**对于Online-Bourgain的修改**

在有向图中找出估计的下限的算法对于无向图来说是一个次要的修改。我们在（见算法5）。

**算法5：**

**输入：**图G的两个结点u和v

**输出：**的估计值

1： 获取SKETCH[u]和SKETCH[v]

2： 对于每一个sketch中的距离集合对应集合*S*，提取，和，。

3 ：计算并找到距离这个量的最大值对应的所有集合用于计算sketch根据出边和入边BFS。

4：以上两个步骤的最大值作为返回值

区别是当我们考虑时，我们用它与0之间的最大值而不是绝对值。这是至关重要的，因为有向图的距离不满足标量的对称属性。在无向图中我们不用去考虑这些，因为在每个节点和每个*S*之间总会有一条路径的并且距离是对称的。然而，在有向图中，可能没有一条路径从（到）一个节点到（从）一个集合（因为输入图不需要强连通）。因此，我们正在减去的数量可能会变成∞（所以以最大消除来获得负值）。注意我们不需要去担心距离变成正无穷（这可以根据对下限的观察的正确性得到验证）。另一个微小的差异是我们这里计算了两个量，和。这只是给了我们一个更强有力的下限，我们将表明，这些量都分别是下限。

*观察4.1 由得到的估计距离 是实际距离的下限。*

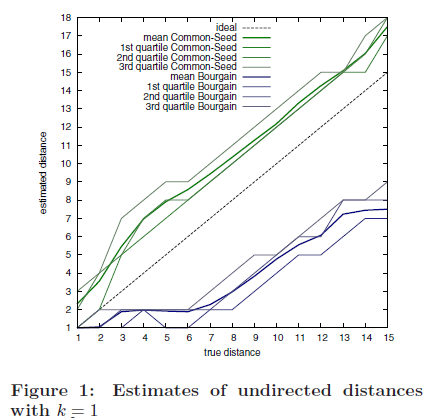
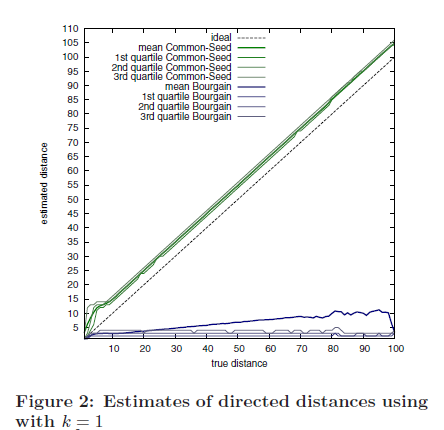
证明。我们需要两个步骤来展示三角不等关系。因为我们是与0比较取最大值，足以证明和。这里保持有向距离是重要的。重新排列，这些遵循三角不等式，对于每一个集合*S，*采用的观察3.3描述的参数顺序。因为我们通过某些集合获得了最大值来计算，这是的一个下界。

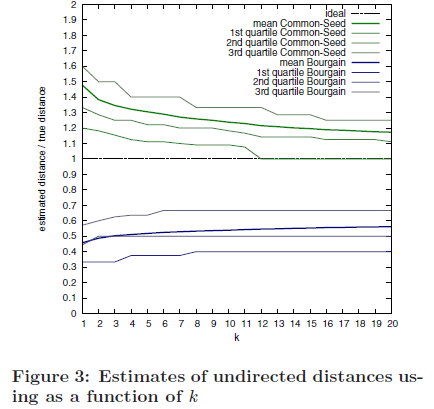
在下面的章节中，我们要比较对于大型网络图算法的性能。

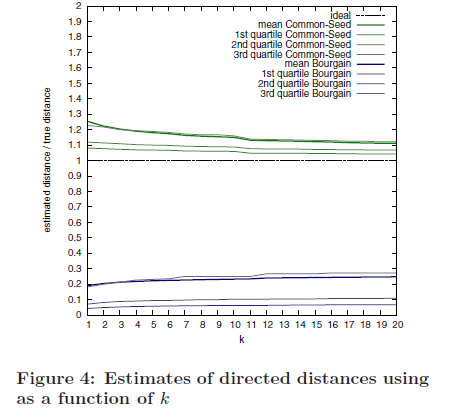
* 1. **实验**

我们的实验是在一个由一个大的爬虫的得到的网络图进行的。这里我们展示了一些基本的数据。这个爬虫是在2002年进行的，是从[www.yahoo.com](http://www.yahoo.com)开始广度优先搜索爬取的结果。我们在这个爬虫的前端进行了我们的实验。爬取的网页数量是65,581,675，不重复的URL数量是419,545,168。这意味着有五倍数量的节点是没有被爬取的“边缘”作为被探索图的一部分。图的总边数为2,371,215,893。爬取的页面平均出度是36.16，所有页面（无论是否被爬取）的平均出度是5.65。最大的出度和入度分别是27,764和1,402,576。

为了在一个给定图中评估我们的算法我们随机采样了100个节点。对于每个节点v，我们计算其到其他所有节点的距离。在有向图的情况下，我们计算从所有节点u到v的距离（这里如果没有从u到v的路径距离为∞）。我们按照距离把u分组，然后随机选择10个节点从每组中，即对于每个给定的距离。这样对于每个给定的图，我们可以得到一个三元测试集。





为了进行我们的实验，我们用Scalable Hyperlink Store[18]，一个跨多个SHS服务器的分布式系统，让每个服务器在主存中保存图的一部分。我们的实现由线下阶段和线上阶段组成。在这个线下阶段，我们选择尺寸以指数增加的种子集，然后计算图中每个节点之间的距离和它最近的种子用Dijikstra算法。对于每个种子集，针对图的每个节点，我们输出了一个由seedid, distance对组成的临时文件。在线下阶段结束时，我们合并所有的临时文件到单个sketch文件，其由每个节点的log n对数据组成。在有向图中，我们运行线下阶段两次——一次计算从种子到节点的距离，一次是节点到种子。为了实现ONLINE-COMMON-SEED，我们重复这个线下阶段k次。最后，我们合并这k（或者2k）个文件成单个文件。

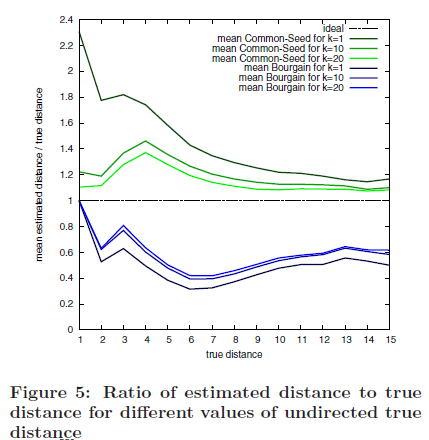
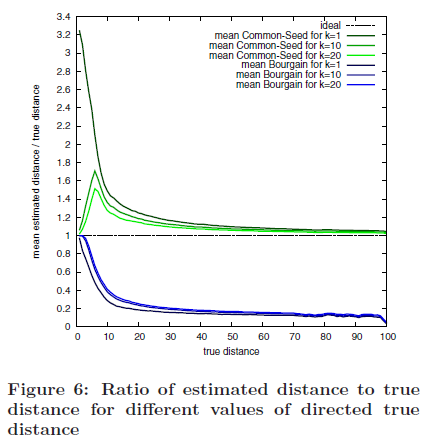
在线上阶段，我们从磁盘上读取所有源和目的节点的向量seedid，distance。所有每次查询涉及两个磁盘寻道。由于一个磁盘寻道花费几个毫秒时间而随后sketch过程仅仅花费几个微秒，我们的算法同Thorup和Zwick的算法一样快，尽管从理论复杂性来看，他们的算法在在线上计算上有更好的时间界限，对于一个的近似需要的时间然而我们的需要。

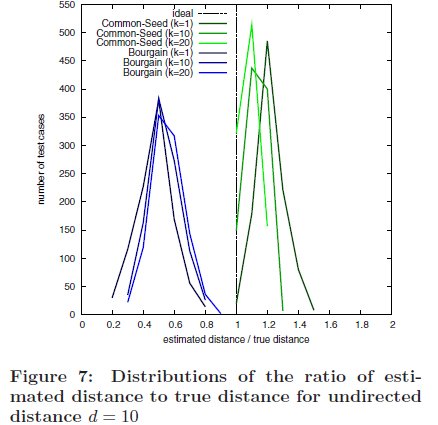
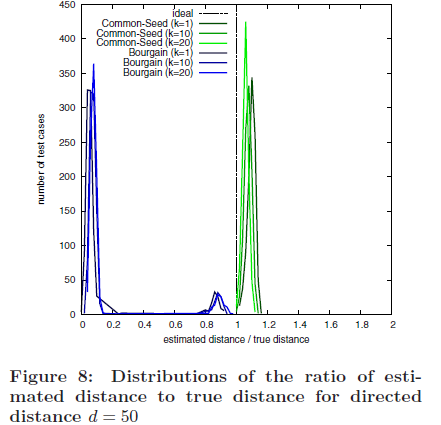
**单种子集采样（k=1）**

在我们的数据集中，实际上成对的无向距离在1到15之间变化。我们的样本对基本覆盖了这些距离，然后查询它们的sketch来找出估计距离。这个有向图不是强连通的。但是对于每对存在有向路径的节点，它们之间的距离是在1到100的范围内。

图表1分别描绘了由算法ONLINE-COMMON-SEED和ONLINE-BOURGAIN得到的估计。在所有的绘图中，x轴表示采样对儿的实际距离，y轴表示估计距离。因此，直线x=y表示对实际距离完美的预测。我们展示了对于平均数，中位数的曲线，第75个百分点和第25个百分点。

同样的对于有向距离的描绘见图表2。我们注意到每个尺寸只有一个被采样的集合用来计算sketch（即k=1）。回想一下，由Online-Common-Seed返回的距离是基于从两个查询节点的sketch中找出公共的节点。另一方面，下限的算法ONLINE-BOURGAIN是基于计算两个查询节点到被采样的集合的距离。首先考虑对无向距离的绘图。注意在图表1中由ONLINE-COMMON-SEED产生的三个四分位是非常接近直线x=y。事实上，即使在真实距离的最大值15附近，我们可以得到大约18的上限在第75个百分位。这表明一个1.2的近似比。考虑到以前的方法仅能保证因子的近似（有时是大的常数）并不能很好的扩展，所有这是一个非常好的近似。在同样的图表中，我们可以看到由ONLINE-BOURGAIN产生的下限也是相当不错的（稳定增长）。然而，比起上限更杂，更弱的保证，大约15/7，就是2.14的近似。





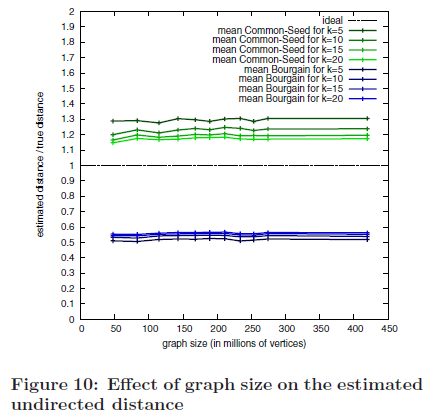
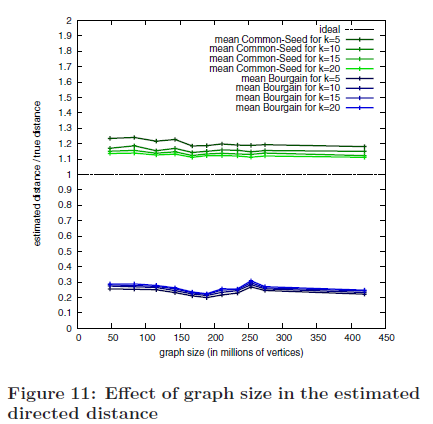
在有向距离的情况中（对于每个log n尺寸重复一次集合采样），图表2就展示了由Online-Common-Seed和Online-Bourgain产生的估计值的巨大差距。虽然ONLINE-COMMON-SEED产生的结果1.05的近似非常好，但是ONLINE-BOURGAIN的表现却非常糟糕。这是由于采样集合的数量不足以捕获许多节点对之间的有向路径。事实上，这是非常困难地去捕获有向路径，有可能很少定向路径。在下面一组实验中，我们研究大量的sketch对我们算法性能的影响。

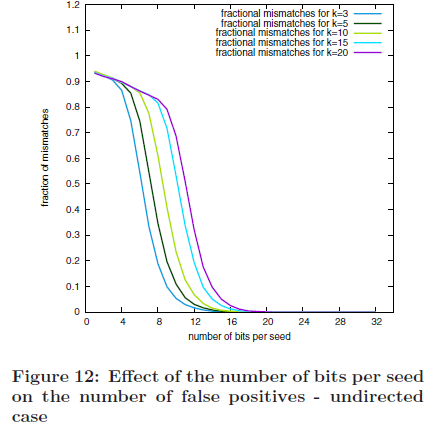
**用更大量的sketch来改善边界**

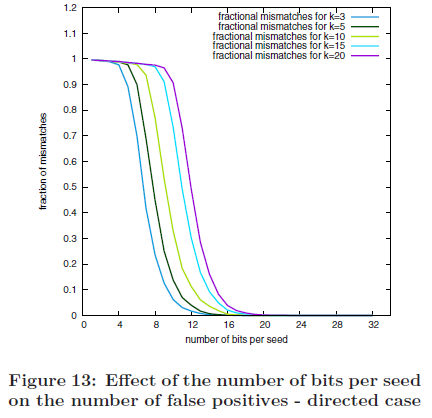
我们可以通过附加的采样来改善在无向距离估计时的巨大差异。我们构建k=20的集合，每一个log n尺寸，用它们来构建每一个节点的sketch。希望能捕获查询节点与集合之间更多的有向路径（这些就在sketch中），因此sketch中集合的总数已经从变成了。这反过来又产生了对两个查询节点之间有向距离的更好的估计。使用了更多的采样集计算的边界在已经描绘在了图表3和图表4。我们并没有绘制实际的的估计距离，取而代之的是我们绘制了估计距离对实际距离的比值作为y轴。因此，该值作为上限将在理想情况y = 1线的上面，并低于该线的相应的下限。

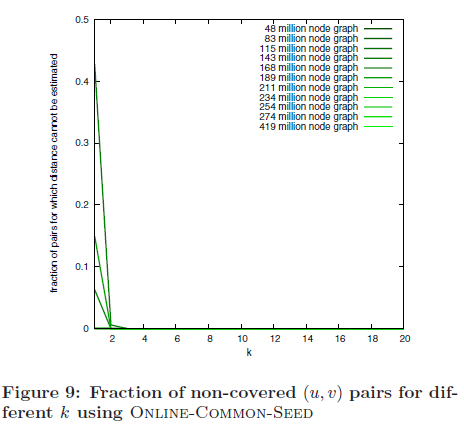
这个采样结果大幅提高了上限。这三个变量——中位数，第75个百分位和第25个百分位开始随着采样数量的增加而靠近y=1。注意上限的估计也由对于k=1的1.6近似值减少到了对于k=20的1.25的近似值。在线下阶段提供足够采样的情况下，这展示了算法ONLINE-COMMON-SEED的成功在计算有向距离估计时有一个高的准确度。另一方面，由Bourgain的算法得到的下限仍然不能与ONLINE-COMMON-SEED相提并论，其在k=20下只有1.85的近似距离值。

另外，在上述实验中，我们计算所有距离上比率的四分位值。为了形象化我们的算法对于任何给定距离的有效性，我们分别对于无向和有向的情况在图表5和图表6中绘制了对实际距离的估计值的中位数。另外，我们考虑k的三个值1,10和20，也说明了采样次数的影响。正如这些图表所展示的，上限的准确性是随着真实距离的增加而增加的。这是因为在一个更小的种子集中两个节点有一个共同种子的可能性是低的。这种趋势在下界处并没有被观察到，事实上，下限的比率在大距离中其值接近为0。第二观察是，样品的数量在有向的情况下也是有助于改善距离估计，尽管更多的是在上限的情况下。







我们注意到当样品的尺寸为k=10时，对于实际距离为10以上的会有一个非常好的近似。这表明，我们可以使用相当少量的样本来从Online-Common-Seed中获得一个良好的性能。为了说明在距离估计中错误的传播，我们对于给定距离计算了其估计距离的分布。图表7和图表8就展示了错误分布的下限和上限，我们分别在无向和有向的情况中选择了d=10和d=50。这些值来自每种情况的中间值。然后，我们再一次在k：1,10和20三个值情况下进行我们的实验。正如我们从图表中看到的，对于下限没有随着k值改变许多，有一个大约0.5（2的近似值）的稠密锋值。此外，k的值不影响concentration的峰值。另一方面，对于由ONLINE-COMMON-SEED产生的值，我们观察到两个趋势。首先，concentration的峰值随着k增加。第二，随着k的增加，concentration值朝着理想值1移动。

在有向距离的情况下，在concentration为1.05左右ONLINE-COMMON-SEED能产生更好的分布，然而ONLINE-BOURGAIN产生的concentration接近于0.1。我们可以看到随着k值增加concentration超理想值1微微移动，这展示了更多样品数量的效果。

在更早的实验中，我们忽略了所有我们的算法不能找到路径的节点对。在无向的情况下没有这样的节点对，因为我们的图是联通的。为了表明在有向的情况下没有如此的条目，我们计算对于不同的k值采样的对中估计无穷距离时导致算法失败的节点对的分值。图表9说明了随着采样数量的增加未覆盖的节点对分值的趋势。从图表中可以看到随着k轻微增加到2以上，这个值快速地跌到了0。这个趋势几乎在所有的图中可以被观察到。结合图表4展示的结果，我们的算法发现了一个对于实际有向距离良好的近似在任意给定节点对之间。

**图尺寸的影响**

在另一个实验组中，我们通过选择不同的广度优先搜索爬虫的前缀来变动图的尺寸，并在一个节点对的采样集上估计有向和无向距离。具体而言，我们考虑了在4.7千万个节点到4.19亿个节点之间的图尺寸。对应的边的个数从1.71亿到23亿。图表10和11就说明了图尺寸分别对于估计有向和无向距离的影响。虽然是在无向的情况下估计距离没有明显的影响，但是Online-Common-Seed在更大尺寸的图中的表现倾向于更好。对这种现象的一个解释是，在更大的图中节点之间可能有更多的路径，因此能被种子集合捕获。

**种子尺寸的影响**

到目前为止，在我们的sketch计算中，我们已经使用了32位唯一的节点标识符来表示种子。现在，我们将探究的种子的有损压缩的影响。我们散列32位种子表示以少的表示，说b，比特。注意这些种子在ONLINE-BOURGAIN上是没有用的，因此算法性能是不受影响的。在算法ONLINE-COMMON-SEED中，我们用在sketch中的种子来标识给定节点对的sketch中公共的种子。用更少的位数哈希种子ID以标识，在这个过程中可能会产生种子误报。事实上，两个种子散列到同样的值的可能性为，由于跨两个sketch的种子中不同节点对的数量增长是k的二次方，所有最终的可能性是 。注意，因为这种误报，算法不一定返回一个上限的真实距离。不出所料，我们观察到，对于k和b的小值，我们得到在上限的估计中合理的准确性，即我们看到了误报的一小部分。图表12和13展示了对于k和b的不同值，这一部分的错误匹配。在无向的情况下，相比有向我们需要较少的比特位来达到同样的精度。然而，随着k的增加，为了降低误报，位数也要随着误报的概率（正比于）增加。我们观察到当k=3, 每个种子12位，错误匹配的部分几乎可以忽略。另外，对于k=3在估计距离中错误也是很小的（见图表3和4）。这个sketch的尺寸能被计算得到，其中来自于节点集合的数量，s是每个种子的比特位数。附加的8位用来存储距离。让k=3，s=12和种子集合的数量为32，分别对于无向图和有向图，我们可以得到sketch的尺寸为240字节和480字节。

* 1. **结论**

我们展示了一个通过获取sketch来支持高效距离查询的算法。在线下阶段计算sketch，然后在线上阶段通过查询任意对节点的sketch来估计它们之间的距离。虽然对于这个问题我们做了理论工作，但是还没有办法扩展到网络图。我们提出了一种算法，通过我们的算法Online-Common-Seed可以估计近似真实的距离，并提供了强大的理论保证。此外，虽然所有先前建议的算法，做了在无向图中估计距离，不过我们的技术也可以被扩展到在有向图中估计距离。我们 比较了我们的算法与Bourgain众所周知的算法基于在低维标量空间中嵌入图。结果证明由Bourgain的算法给出的下限是显著的减弱的。这是无法预测的没有精确的有向结果。对于无向图，而这是无法在精确度方面符合我们的上限算法，它确实给一个相当不错的预测。

我们进行本文中提到的所有算法的扩展实验，并与采样节点对的真实距离做比较。这个实验是在一个大型网络图的爬虫上运行的。据我们所知，这是第一个关于distance oracle应用的实际数据在这个规模上具有实际意义的工作。我们计划运行这些算法在其他大图中，例如Instant Messenger t图。

**致谢**

这里我们想要去感谢一下Ittai Abraham帮助我们给我们的距离估计算法一个更加强健的理论保障的证明

* 1. **引用**

[1] Yair Bartal. On approximating arbitrary metrices by

tree metrics. In STOC, pages 161–168, 1998.

[2] Surender Baswana. Streaming algorithm for graph

spanners - single pass and constant processing time

per edge. Inf. Process. Lett., 106(3):110–114, 2008.

[3] Jean Bourgain. On Lipschitz embeddings of finite

metric spaces in hilbert space. Israel Journal of

Mathematics, 52(1-2):46–52, 1985.

[4] Andrei Z. Broder. Identifying and filtering

near-duplicate documents. In CPM, pages 1–10, 2000.

[5] Andrei Z. Broder, Steven C. Glassman, Mark S.

Manasse, and Geoffrey Zweig. Syntactic clustering of

the web. Computer Networks, 29(8):1157–1166, 1997.

[6] Edith Cohen, Eran Halperin, Haim Kaplan, and Uri

Zwick. Reachability and distance queries via 2-hop

labels. In SODA, pages 937–946, 2002.

[7] Reuven Cohen, Pierre Fraigniaud, David Ilcinkas,

Amos Korman, and David Peleg. Labeling schemes for

tree representation. Algorithmica, 53(1):1–15, 2009.

[8] Thomas H. Cormen, Charles E. Leiserson, Ronald L.

Rivest, and Clifford Stein. Introduction to algorithms.

MIT Press and McGraw-Hill, 24.3:595–601, 2001.

[9] Camil Demetrescu, Andrew V. Goldberg, and

David S. Johnson. Implementation challenge for

shortest paths. In Encyclopedia of Algorithms. 2008.

[10] Wei Dong, Moses Charikar, and Kai Li. Asymmetric

distance estimation with sketches for similarity search

in high-dimensional spaces. In SIGIR, pages 123–130,

2008.

[11] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar.

A tight bound on approximating arbitrary metrics by

tree metrics. J. Comp. Syst. Sci., 69(3):485–497, 2004.

[12] Joan Feigenbaum, Sampath Kannan, Andrew

McGregor, Siddharth Suri, and Jian Zhang. Graph

distances in the streaming model: the value of space.

In SODA, pages 745–754, 2005.

[13] Fran¸cois Fouss, Alain Pirotte, Jean-Michel Renders,

and Marco Saerens. Random-walk computation of

similarities between nodes of a graph with application

to collaborative recommendation. IEEE Trans. Knowl.

Data Eng., 19(3):355–369, 2007.

[14] Cyril Gavoille, David Peleg, St´ephane P´erennes, and

Ran Raz. Distance labeling in graphs. J. Algorithms,

53(1):85–112, 2004.

[15] Andrew V. Goldberg. Point-to-point shortest path

algorithms with preprocessing. In SOFSEM (1), pages

88–102, 2007.

[16] Michal Katz, Nir A. Katz, Amos Korman, and David

Peleg. Labeling schemes for flow and connectivity.

SIAM J. Comput., 34(1):23–40, 2004.

[17] Jiri Matousek. On the distortion required for

embedding finite metric spaces into normed spaces.

Israel Journal of Mathematics, 93(1):333–344, 1996.

[18] Marc Najork. The scalable hyperlink store. In

Hypertext, pages 89–98, 2009.

[19] Mikkel Thorup and Uri Zwick. Approximate distance

oracles. J. ACM, 52(1):1–24, 2005.

[20] Luh Yen, Marco Saerens, Amin Mantrach, and

Masashi Shimbo. A family of dissimilarity measures

between nodes generalizing both the shortest-path and

the commute-time distances. In KDD, pages 785–793,

2008.