**Bring order into the samples: A novel scalable method for influence maximization**

Wang, X., Zhang, Y., Zhang, W., Lin, X., & Chen, C. (2016).. *IEEE Transactions on Knowledge and Data Engineering*, *29*(2), 243-256.

**Abstract**

本文提出了一种新的基于底边k草图的RIS框架，即BKRIS，它将样本的顺序引入到RIS框架中，而不是花费大量的精力来确定一个较小的样本容量。通过应用草图技术，我们可以得到早期终止条件来显著加快种子集的选择过程。此外，我们提供了一种经济有效的方法来寻找合适的样本量来约束返回结果的质量。此外，我们提供了一些优化技术，以降低生成样本顺序的成本，并有效地处理最坏的情况。我们在10个真实世界数据集上验证了该方法的有效性。与IMM方法相比，BKRIS可以达到2个数量级的加速，影响范围几乎相同。在拥有18亿个边的最大数据集中，BKRIS可以在1.3秒内返回50个种子，在36.6秒内返回5000个种子。IMM分别用了55.32秒和3664.97秒

**Introduction**

最近，Borgs等人开发了一个优雅的框架——反向影响抽样(RIS)来解决影响最大化问题。

产生样本的过程可分为两个阶段: 1)随机选择一个节点v，2)在一个采样实例g中获得节点v的反向可达集合Rg(v)。很容易看出，第二阶段的成本比第一阶段要大得多。为了解决RIS框架下的影响最大化问题，我们首先选择一个样本集合R，然后迭代地选择与R具有最大边缘重叠大小的节点

节点选择的时间复杂度与样本的大小成正比，因此，样本容量的大小直接决定了RIS框架的效率和有效性。

Example1：给定一组样本，我们可以在样本上建立二部图，其中的边表示G中节点与样本节点之间的可达性关系。假设我们有100个样本。图1(a)是基于样本构建的相应的二部图。上层由G中的节点组成，下层由采样的节点组成，那么u1是第一个选择的节点，因为它覆盖了最大数量的样本，u3是选择的第二个节点，因为它具有最大的边缘覆盖。

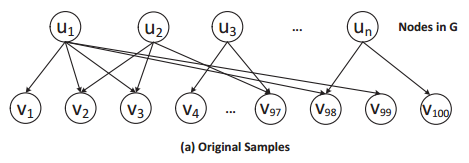
假设我们有一个给所有样本分配总顺序的oracle。这个顺序可以提供一个属性，如果一个节点覆盖了前面排列的更多的样本，它就更有可能产生更大的影响。在形式上，oracle提供了一个覆盖需求r。它保证了如果我们按照顺序遍历样本，第一个覆盖r个样本的节点将是覆盖范围最大的节点。

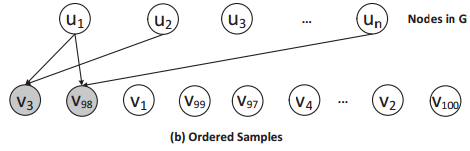
如图1(b)所示，我们根据oracle和需求r = 2对图1(a)中的抽样节点进行排序。将2个样本v3和v98具体化后，我们发现u1是第一个覆盖2个样本的节点。对于下面的示例，如果它们被已经选择的节点覆盖，**我们将删除它们**。例如，我们将在物化它们时丢弃样本v1和v99，因为u1可以到达它们。随后第二个被选择的节点是u3因为他可以到达v97和v4。

有了这样一个oracle，我们可以在节点选择阶段实现可能的提前终止，而不需要实例化所有的样本。

**贡献**：1.我们提出了BKRIS框架，它通过引入基于底部k草图的样本顺序来加速RIS框架。

2. 提出了一种利用小世界性质导出一个充分合理的大样本容量的有效方法。





**2. Background**

**2.2 preliminaries**

**2.2.1 Bottom-k Sketch**

在本节中，我们将简要介绍bottom-k草图，在作者提出的BKRIS框架中，用于获取提前终止的统计信息。底部k草图用于估计多重集中不同值的数量。设N个不同的点均匀分布在(0,1)上，则任意两个相邻点之间的期望距离为。给定一个多重集，一个真正随机的哈希函数h，A中的每个不同的值vi被哈希到(0,1)。A的bottom-k草图包含k个最小的哈希值。

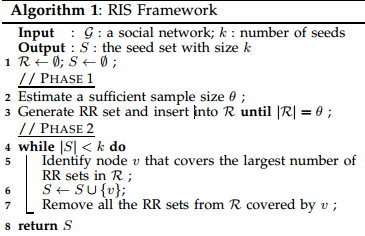
**2.2.2 The RIS based Approaches**

首先讲解如何构造一个图G的实例g。在IC模型中，每条边以1-p(u,v)的概率进行删边。在接下来的表述中，一个RR集合代表一个样本（sample）。RIS框架的思想是，给定一个RR集合R，如果一组节点频繁出现在R中，则更有可能产生较大的影响扩散。

方程(2)是σ(S) 的一种无偏估计，中F (S, R)是种子集合S在覆盖的RR集合的数量。



因此，我们可以生成大量的RR集，并找到覆盖最大RR集的种子集。基于这种直觉，RIS框架可以总结为算法1。



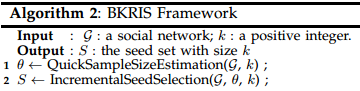
**RIS框架由两个阶段组成**。在第一阶段,它试图获得一个样本大小θ,将足以保证返回的种子集的质量,然后返回一组R和θ个R集。在第二阶段，它经历k次迭代。在每个迭代中，它将具有最大边际覆盖率的节点添加到种子集。

Borgs等人使用生成RR集期间访问的边数来确定样本大小何时足够。IMM[11]是先进的方法,利用鞅方法进一步减少λ和重用这些样本的大小，因此，它可以增量地生成RR集并估计一个更小的OP T下界，直到满足停止条件。

**3** **BOTTOM-K BASED RIS FRAMEWORK**

**3.1 motivation and general framework**

现在研究的重点是用新的技术来减少样本量。即使IMM可以重用样本，并且与TIM/TIM+相比，可以显著减少样本大小，但是当k增加时，IMM的开销仍然很大。所以作者提出了BKRIS方法。它的总体框架如下：



Quick Sample Size Estimation：作者的第一阶段的目标是快速估计足够样本量θ。样本容量的充分性定义如下。如果我们进行RIS节点选择算法与θ样本, 它应该返回一组种子1−−1 / eǫ近似比例有高概率。为了实现这一目标，我们开发了一个有效的算法，它返回一个OP T的下界。将下界代入IMM中的方程，我们可以得到所需的RR集的数量。

Incremental Seed Selection：在种子选择阶段，我们使用bottom-k sketch，在样本排序的基础上，逐步实例化样本，构造二部图。

**3.2 Quick Sample Size Estimation**

Sample Size Requirement.：根据第1节的例子，给定一个规则，我们可以找到种子集，而不需要实例化所有的样本。因此,样本容量θ应该满足以下三个要求。

1. 样本量不能太大。否则，在最坏的情况下，它将运行缓慢

2. 样本量应足够。这里的充分性意味着样本容量不应该小于，（α,β是l, k, n的函数）

3. 样本容量的计算是有效的，否则将成为整个算法的瓶颈。

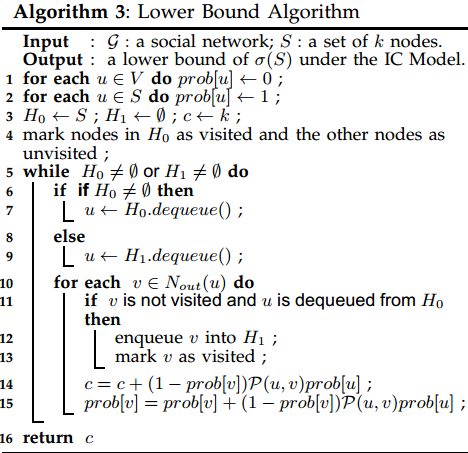
为了满足上述要求，我们需要有效地获得一个OPT的下界，并将其代入IMM所使用的方程中。注意，我们只是在IMM中使用了样本量方程，而我们在这里的贡献是提供了一个具有成本效益的方法来获得OP T的下界。

在本文中，我们利用小世界性质推导了OPT的下界，也就是说，我们试图通过探索一组k个节点的有限步长来获得下界。该性质被许多启发式方法广泛用于求解影响最大化问题。但在我们的问题中，我们不能直接采用这些方法，因为这些方法更侧重于选择k个有影响力的节点，而不是有效地获得或估计最大影响力扩散的下界。在本文中，我们只利用k个节点的2跳邻居。

具体来说，我们探索了广度优先搜索(BFS)方式的k个节点，我们一层一层地访问节点，每个节点只访问它的外部邻居一次。这种策略是快速的，但与通过2跳邻居计算出的确切影响相比，它可能会导致下降。所以它是k个节点影响的下界，它显然是一个OPT的下界。

**Obtain a Lower Bound of OPT under the IC Model**：

在IC模型下的方法细节在算法3中给出，算法的输入包括一个社交网络G和一个选择的节点集S，节点数为k。首先，我们将所有节点的激活概率prob[u]设为0,S中的节点除外，其概率在第1行和第2行中设为1。H0和H1是两个队列。H0存储种子节点，H1存储S的第一层访问的所有节点。c是一个存储计算出的累积影响的计数器，c最初设为k。在第4行，我们将H0中的所有节点标记为已访问节点，而将其他节点标记为未访问节点，该标记用于避免将同一节点放入H1多次。然后我们开始逐个访问H0中的节点，将第一层访问的节点放入H1中。对于每个离开队列的节点u，我们访问它的每个出邻居v，v的累积影响可以计算为，因此，累积影响就是u对v的影响增益。对H0中的节点进行处理后，我们在H1的节点上继续这个过程，直到H1为空。根据IC模型的定义，该算法返回的下界为σ(S),所以它也是一个下界的OP T。



**Choose the Nodes for Calculating the Lower Bound**：为了选择一组k个节点输入算法3，我们使用基于度的启发式方法。我们不使用其他的启发式方法，因为它们的代价太大。

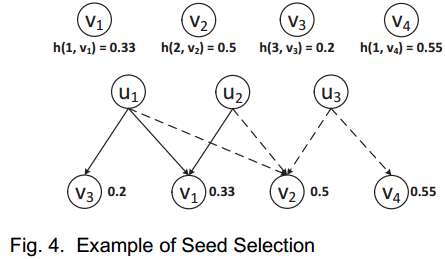
**3.3 Incremental Seed Selection**

在这一节中，我们将介绍使用bottom-k草图的增量种子选择方法，基于样本量θ,我们的目标是找到k个种子覆盖大多数样本，此外，我们还介绍了几种优化方法，以加速搜索过程并减少搜索空间。我们首先介绍选择第一粒种子的过程，然后扩展到选择k粒种子

**3.3.1 Select the First Seed**

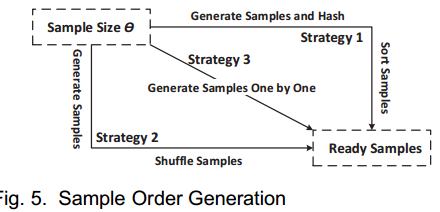
考虑到样本容量θ,我们在G中随机选择θ节点，，i是这个样本的id，因为我们可以对同一个节点进行多次采样。请注意，我们没有在这里实现样本。对于G中的每个节点，我们维护一个用0初始化的计数器，对于每个样本(i, v)，我们随机生成一个散列值h(i, v)∈(0,1)，并按哈希值的递增顺序对样本进行排序。注意，对于sample (i, v)，我们可以使用sample的id i作为键来生成散列值，对于不同的样本，它们的id是不同的，因此即使我们对相同的节点采样，也会生成不同的哈希值。接下来，我们依次对样本进行处理，设(i, v)为当前样本。我们首先检索RR集R(i, v)，对于每个节点u∈R(i, v)，我们将u的计数器增加1，当一个节点的计数器到达bk(预先定义的阈值)时，我们将它作为第一个种子返回。如果处理完所有样本后没有节点的计数器到达bk，则返回计数器最大的那一个作为第一种子。

Example 4：



Optimization of Sample Order Generation：

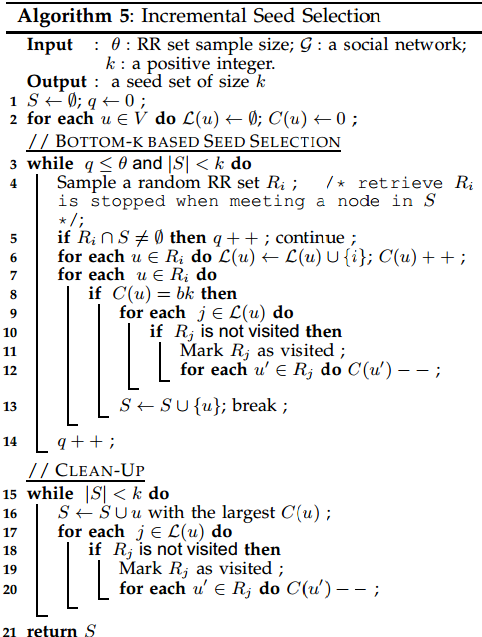
在我们的问题中，我们只打算选择第一个种子，而不是估计节点的覆盖率，因此我们只关心基于它们的哈希值的样本的顺序。很容易验证，如果删除了样本的哈希值，只保留样本的顺序，那么选择第一个种子的过程仍然是正确的。因此，我们可以将策略1替换为图5中的随机打乱选择的样本。除此之外还可以一个一个样本进行生成。



我们只需要生成样本在需要的时候,而不是使用O(θ)空间采样节点提前。在这种情况下，我们节省了空间和时间。

**3.3.2 Select Subsequence Seeds**

基于选择第一粒种子的思想，我们可以很容易地将其扩展到选择k粒种子。一般的想法是，对于每个节点u，我们维护一个列表，记录到目前为止它可以访问的样本，并为该列表提供一个计数器。对于下面的每个样本(i, v)，如果它的RR集包含种子集合中的节点，我们就丢弃它。在这种情况下，剩下的条件相当于选择第一颗种子。算法5描述了选择k个种子的细节。



该算法可分为两个步骤。第一个是基于k-bottom的种子选择，第二个是一个清理程序，用来处理我们不能使用第一个程序选择k种子的情况。q是一个计数器，用于计算采样的RR集的数量, L(u)记录到目前为止u到达的样本, C(u)是它对应的计数器。当q小于θ，|S|小于k，继续检索样本。第4行和第5行可以合并为一个流程，即，当我们尝试获得一个样本的RR集时，如果我们遇到S中的一个节点，我们将丢弃这个样本并进入下一个迭代，否则，我们将更新第6行RR集中的计数器和u列表。如果一个节点的计数器到达bk，我们选择它作为下一个种子，并在第8 - 13行从其他节点的列表中删除它所覆盖的所有样本。当我们在第一个过程之后无法选择k种子时，就进入清理阶段，简单地根据第16行中节点的计数器逐个选择节点，选择节点之后，还需要将其他节点的列表和计数器更新。