Efficient Algorithms for Adaptive Influence Maximization

背景

之前许多算法都是在研究非渐进式IM问题时被提出的，即一次性选择k个种子节点，观察它如何影响其它节点。而实际上可以先选择一部分种子节点，观察它影响的节点，然后再在剩余未被影响节点选择一部分种子节点，观察其影响的节点，直到达到选择种子节点的数量为止，即采用渐进式的方式来确定种子节点如何选择。

解决问题

解决了当前研究渐进式选择种子节点算法存在的问题--任何现有的高效算法都无法满足其对传播期望估计的准确性要求。

创新之处

- 提出了一个新颖的通过非渐进IM算法实例化的渐进式贪心框架来解决渐进式IM问题。
- 进一步提出了一个不落后于先进IM算法的非渐进式IM算法EPIC，能够适应于渐进式贪心方法的性能要求。

解决方法

渐进式IM问题

假设在r批次内一共选择k个种子节点，那么每个批次选择的种子节点为$b = k/r$个。渐进式IM问题要求选择r个种子集合$S_1, \ldots, S_r$，并且$S_1 \cup S2 \ldots \cup S_r = S, |S| = k$，以至第$i$个批次选择的$S_i$在选择的概率图$w$（$w$通过$1 - p(u, v)$的概率移除图$G$中的边$(u, v)$生成）上影响传播期望最大。

AdaptGreedy

Algorithm 1: AdaptGreedy

Input: social network G, seed set size k, batch number r
Output: adaptively selected seed sets $S_1, \cdots, S_r$

1 $b \leftarrow k/r$;
2 $G_1 \leftarrow G$;
3 **if** $r = k$ **then**
4 $\quad c \leftarrow 1$;
5 **else**
6 $\quad c \leftarrow 1 - 1/e$;
7 **for** $i = 1$ **to** r **do**
8 $\quad$ Identify a size-b seed set S_i from G_i, such that the expected spread of S_i on G_i is at least $c - \xi_i$ times the largest expected spread of any size-b seed set on G_i;
9 $\quad$ Observe the influence of S_i in G_i;
10 $\quad$ Remove all nodes in G_i that are influenced by S_i, and denote the resulting graph as G_{i+1};
11 **return** $S_1, \cdots, S_r$

line 3的if条件是区别两种情况，即非渐进式和渐进式。

算法本身：

1. 循环批次 r 次，每次选择 b 大小的种子节点，并使其得到的影响传播至少是最大影响传播的 $c-\xi_i$ 倍
2. 观察 S_i 在剩余图 G_i 上的影响传播
3. 删除图 G_i 中被 S_i 影响的节点，生成新图 G_{i+1}
4. 最终返回得到的 $S_1,\dots,S_r$

EPIC

EPIC和SSA算法相似：

1. 算法开始于一个小数量的RR集合 $\mathcal{R}$
2. 通过最大集合覆盖算法选取RR集合率最大的种子集合，迭代增加RR集合数量的大小，直到得到的解满足确定条件
3. 返回满足条件的解

不同之处在于增加RR集合的数量时，EPIC每次增加和上次RR集合等数量的RR集合。

近似保证

Vaswani and Lakshmanan提出的方法，允许每次期望传播的估计有一个误差，对于种子集合在G_i上的估计$\tilde{\mathbb{E}}[I_{G_i}(S)]$满足：

$$c^{\perp}\mathbb{E}[I_{G_i}(S)] \le \tilde{\mathbb{E}}[I_{G_i}(S)] \le c^{\mathsf{T}}\mathbb{E}[I_{G_i}(S)]$$

但当$\mathbb{E}[I_{G_i}(S)]$很小时，允许的误差也很小，这使得计算$\tilde{\mathbb{E}}[I_{G_i}(S)]$具有挑战性。该方法作者提出以牺牲准确率换取效率，但无法得到非平凡近似保证。

作者的AdaptGreedy框架大致和Vaswani and Lakshmaman提出的方法相似，只是在误差方面，作者为每个种子集合定了一个绝对误差$\xi_i OPT_b(G_i)$，$OPT(G_i)$是在G_i上最大的影响传播。作者证明了它提供了一个非常强的近似保证：

$$\begin{cases} 1-exp(\xi-1), & if\ b=1 \\ 1-exp(\xi-1+1/e), & otherwise \end{cases}$$

其次，作者通过设计一个非渐进算法保证了通过这个算法选取种子集合，达到想要的近似保证，并通过理论证明其可行性。

先前的研究

- Golovin et al.采用每次只选择一个种子节点的方法，得到了 $1-1/e$ 的近似率
- Vaswani and Laskshmanan采用每次选择多个种子节点的方法，得到了 $1-exp(-\frac{(1-1/e)^2}{\eta})$, $\eta>1$ 的近似率。存在任何现有的高效算法都无法满足其对传播期望估计的准确性要求的问题。
- Chen et al.研究目标在于最小化选择种子节点的成本。
- Seeman et al.，Horel et al. 和 Badanidiyuru et al.考虑“adaptive seeding”的影响力最大化，实现思路有所不同。

实验

数据集

Dataset	n	m	Type	Avg. degree
NetHEPT	15.2K	31.4K	undirected	4.18
Epinions	132K	841K	directed	13.4
DBLP	655K	1.99M	undirected	6.08
LiveJournal	4.85M	69.0M	directed	28.5
Orkut	3.07M	117M	undirected	76.2

实验环境：Linux machine with an Intel Xeon 2.6GHz CPU and 256GB RAM

H3 算法

- AdaptIM-1 和 AdaptIM-2（基于不同的误差理论提出的算法）
- D-SSA 和 IMM

参数设置：

- 传播模型使用IC模型
- 传播概率 $e=(u,v)=1/d_{in}(v)$
- 选择 k 个种子节点， r 个批次，每次 b 个种子节点，对于 b 的设置，固定 $k=500$ ，$b\in 1,2,5,10,20,50,500$ ，对于 k 的设置，固定 $r=50$ ， $k\in 50,100,200,\ldots,500$

H3 运行时间

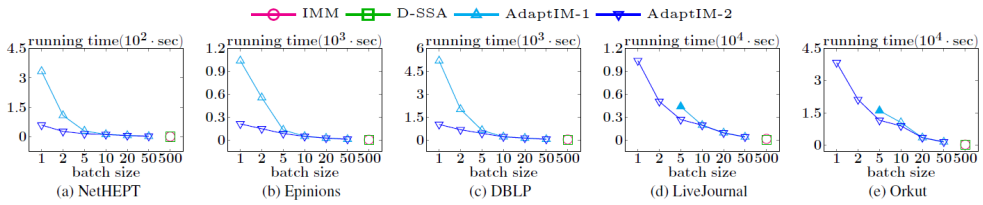

Figure 3: Running time vs. batch size

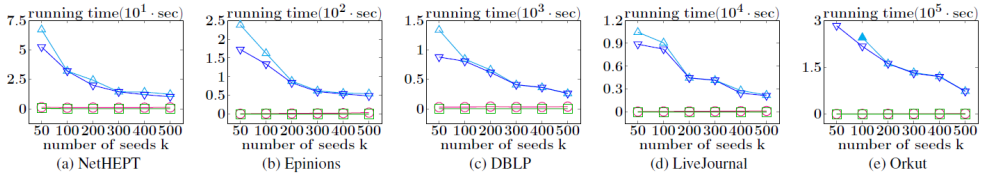

Figure 4: Running time vs. seed size

- 图3是固定 $k=500$ ， r 在范围内变化，各个算法运行时间的变化情况。因为IMM和D-SSA只能适应于非渐进IM，所以运行时间为 $r=500$ 时所显示的。而作者提出的两种算法，随着批次增加，运行时间随之减少。因为渐进式是分批次的处理，所以批次越多，处理时间越长。
- 图4是固定批次 r ，让 k 变化，应该是没有测试非渐进式算法，只是比较了作者提出的两个算法。

H3 影响传播

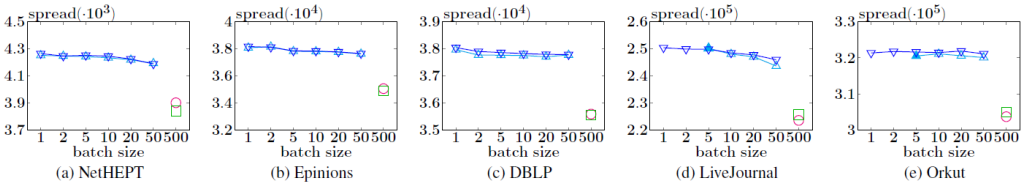

Figure 5: Spread vs. batch size

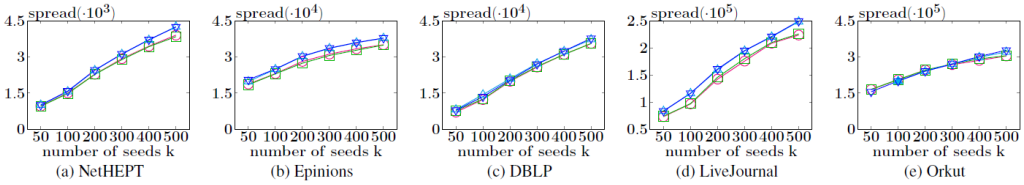

Figure 6: Spread vs. seed size

- 图5是固定 $k = 500$ ， r 在范围内变化，比较各个算法的影响传播大小。因为IMM和D-SSA只能适应于非渐进IM，所以影响传播为 $r = 500$ 时所显示的。可以明显的看出，渐进式算法得到的影响传播要高于非渐进算法很大的margin。
- 同样，图6是固定批次 r ，让 k 变化，AdaptIM在不同数量种子节点的设置下都比UNAdaptIM算法影响传播要高。

H2 思考

1. 该渐进式IM问题是将 k 个种子节点平均分成 r 分，每个批次选择 $b = k/r$ 个种子节点，是否可以考虑不平均分的情况，比如是满足某个序列。
2. 无论是自激活还是渐进式，应该加上成本条件约束，以最大化收益为目标才更有意义，是否可以将问题变为利益最大化问题。
3. 结合之前基于社区的影响最大化。它的判断指标是，如果社区内一半节点被影响那么该社区节点全部被影响。那么考虑渐进式激活的方式，每个批次记录每个社区所被激活节点的数量，当满足社区一半节点被激活时，直接将该社区从图中移除，以加快算法执行速度。