Revenue Maximization by Viral Marketing: A Social Network Host's Perspective

背景

传统的病毒营销问题是在社交网络中确定k个种子用户，从而使受到影响的用户数量期望最大化。这个开销k依赖于有多少被公司打广告，赠送样品和价格折扣而购买他们的产品的初始用户数量。

大部分关于该领域的研究都是假设社交网络结对与推销公司来说是已知的。实际上，社交网络平台属于第三方主机，这些主机会因为自身利益和另外的私人原因保护社交网络图结构的秘密。因此，营销公司会因为缺乏社交网络图信息而不能选择最好的种子集合。

文中假设社交网络主机代替它的客户端即营销公司们进行选择种子集合，而竞争者（公司们）向主机支付费用。每个竞争者包括种子集合大小的开销以及对应每个受影响的用户，他愿意向主机支付的费用。主机要为每个竞争者分配不相交的最优种子集合，使自己得到的收入最大化。这是一个社交网络病毒营销的多竞争者模式。

解决的具体问题

- 解决了第三方主机保守社交网络结构的秘密从而使营销公司不能获得最优种子集合的问题。
- 解决了先前提出的基础的多竞争者病毒营销方法假设不实际从而不能使主机获得最大收入问题。

解决方法

Revenue Maximization

给定一个社交网络$\mathcal{G}=(V,E,P)$，$m\geq 2$个竞争者，收入矩阵A，以及对于每个竞争者C_i的种子用户大小及开销k_i，即$|S_i|=k_i, 1\leq i\leq m$，为每个竞争者找到一个种子集合使得主机的期望收入最大化。表示为：

$$
\begin{gathered}
argmax_{S_1,S_2,\ldots,S_m}\sum_{i=1}^{m}\sum_{u\in V}[A_{iu}\cdot Pr(u,i,S)] \\
such\ that\ |S_i|=k_i\ \forall i\in(1,m) \\
and S_i\bigcap S_j=\phi\ \forall i\neq j;\ i,j\in(1,m)
\end{gathered}
$$

Influence Cascading Models

MCIC

多竞争者独立级联模型的传播影响过程和独立级联模型大致相同，主要有两个区别：

- 如果节点 u 在 C_i 下被激活，它会在 C_i 下企图去激活它的出邻居节点 v 。
- 被激活的节点 v 随机均匀的从其在最近一轮被激活的入邻居中采用一个 C_i 。

每个节点只能通过一个竞争者被激活一次，节点激活状态持续到竞争结束。

K-LT

多竞争者线性阈值模型也被称为K-LT模型，在每个回合有两步：

1. 确定一个新的节点是否被激活，过程同线性阈值模型。
2. 决定被激活的节点是采用哪一个 C 。考虑所有最近一轮被激活的节点 u 和当前一轮被激活的节点 v ， v 将会通过概率 $\frac{p_{uv}}{\sum_u p_{uv}}$ 来确定是否采取与 u 相同的 C 。

每个节点只能通过一个竞争者被激活一次，节点激活状态持续到竞争结束。

之后作者证明了在这两个级联模型下，Revenue Maximization是Np-Hard的，且目标函数是非子模性和非单调性的。

RevMax-C

文章首先分别给出了在MICI和K-LT模型下Revenue Maximization的基于动态规划的联合优化算法。

1. MCIC RevMax-C

- 有向图转为最大影响力有向树

  给定一个连接图$\mathcal{G}=(V,E,P)$和一个种子节点v_r，最具影响力树$T^*=(V,E_{T^*},P)$，$E_{T^*}\subseteq E$是G的边概率乘积最大的有向生成树。表示为：

  $$T^*=argmax_{T\in SpanningTrees(G)}\prod_{(u,v)\in E_T}p_{u,v}$$

  $SpanningTrees(G)$代表G的有向生成树。而找到这样的最具影响生成树可以通过最小负的对数和转化为求有向图的最小生成树问题，表示为：

  $$T^*=agrmin_{T\in SpanningTrees(G)}\sum_{(u,v)\in E_r}-log(P_{u,v})$$

  有向图最小生成树Zhu-Liu算法大致思想：

  1. 找到除根节点外其它节点的最小权值的入边
  2. 如果出现了除根节点外其它的孤立点，则不存在
  3. 之后找到图中所有的环，并将环缩为点，重新编号
  4. 更新其它环外的点到环的权值：

     $g[v][U_k]=min\{gh[v][U_{kj}]-mincost[U_{kj}]\}$

     即环外点v到环内点U_{kj}的权值更新为原本权值减去环内U_{kj}最小入边权值。
  5. 重复以上步骤，直到五环为止。

  重要的是需要一些根节点v_r作为输入，以致图G中所有的其它节点都可以通过v_r到达。因此加入一个虚假的根节点v_r，连接其与G中的所有节点。每一天新加入的边都被设置为一个超低概率，且设置$A_{iv_i}=0$，并令v_r不能被选为种子节点。

- 有向树转为有向二叉树

  转化思想是对于源树每个非叶节点v和它的孩子节点$v_1,v_2,\dots,v_k$，$k\geq 2$，用一颗深度最大为$log k$叶子为$v_1,v_2,\dots,v_k$的二叉树替换v。对于每个新加节点让其收入$A_{iu}=0$，且不能被选为种子节点。对于新加有向边让其概率为1。

- 树形DP求解

  1. 设置一张表$\mathcal{B}$储存离v最近的祖先节点对其激活概率$Pr(v,i,S)$：

     $$Pr(v,i,S)=\begin{cases}0 \quad if\ u\in S_i;\\ \prod_{(u',v')\in Path(u,v)}p_{u',v'} \quad otherwise\end{cases}$$

  2. 设置一张$OPT(v,u,j,[k'_1,k'_2,\dots,k'_m]^T)$表，$v$是树中的任意节点，$u$是离$v$最近的祖先种子节点，$j\in(1,m)$表示$C_j$，$u$是$C_j$的种子节点。填入的信息表示在以$v$为根的子树上，主机最佳分配所有种子集合$S_i,i\in(1,m),|S_i|=k'_i$后取得的最大期望收入。

     $OPT(v,u,j,[k'_1,k'_2,\dots,k'_m]^T)$由计算两个最大化情况得到：

     $$OPT\left(v,u,j,\begin{bmatrix}k'_1\\ .\\ .\\ .\\ k'_m\end{bmatrix}\right)=max\{Case_1,Case_2\}$$

     1. v不是种子节点：

$$Case_1 = max_{k'_1=0}^{k'_1} max \cdots max_{k'_m=0}^{k'_m} \left\{ OPT\left(l(v), u, j, \begin{bmatrix} k''_1 \\ \cdot \\ \cdot \\ \cdot \\ k''_m \end{bmatrix}\right) + OPT\left(r(v), u, j, \begin{bmatrix} k'_1 - k''_1 \\ \cdot \\ \cdot \\ \cdot \\ k'_m - k''_m \end{bmatrix}\right) + A_{j,v} \times \mathcal{B}(u,v) \right\}$$

3. v是种子节点：

$$Case_2 = max_{i=1}^{m} \left\{ max_{k''_1=0}^{k'_1} max \cdots max_{k''_m=0}^{k'_m} \left\{ OPT\left(l(v), u, j, \begin{bmatrix} k''_1 \\ \cdot \\ k'_i \\ \cdot \\ k''_m \end{bmatrix}\right) + OPT\left(r(v), u, j, \begin{bmatrix} k'_1 - k''_1 \\ \cdot \\ k'_i - k''_i - 1 \\ \cdot \\ k'_m - k''_m \end{bmatrix}\right) + A_{j,v} \right\} \right\}$$

时间复杂度：$O(ndm^2k^{2m})$

H5 2.K-LT RevMax-C

1. 求最优种子集合

作者假设主机收益最大化可以由只给一个C选取种子集获得,而这里让$A_u = max_{i\in(1,m)}\{A_{iu}\}$。因此可以通过传统的最优种子集选择问题来解决，选取mk个节点的种子集合，使其主机期望收入最大化：

$$argmax_S \sum_{u\in V}[A_u \cdot Pr_{LT}(u,S)], |S| = mk$$

其目标函数是子模性且单调的，因此可以通过爬山法求解：

Algorithm 1 Hill-Climbing for Optimistic Seed Set Selection

Require: Graph $G = (V, E, P)$, $A_u = \max_i\{A_{iu}\}\ \forall u \in V$
Ensure: Seed set S of size mk
1: $S = \phi$
2: **for** $i = 1$ **to** mk **do**
3: $v = \arg\max_{v\in V\setminus S} F(S\bigcup\{v\})$ // $F()$ is defined in Th. 3
4: $S = S\bigcup\{v\}$
5: **end for**
6: Output S

时间复杂度：$O(mkn + mke + m^2k + mk^m)$

2. 分割种子集

上述得到种子集合后，定义一个新问题：

给定一个mk大小的种子集合和收入矩阵$(A_{iu})_{m\times n}$，分割S为子集$S_1, S_2, \ldots, S_m$，每个S_i大小为k，在K-LT模型下求得主机的最大期望收入。

然后使用动态规划求解：

定义一张表$EXACT(j, [k'_1, k'_2, \ldots, k'_m]^T)$，储存主机分割种子集给$m$个竞争者后的最大期望收入，$1 \le j \le mk\ and\ j = \sum_{i=1}^{m} k'_i$。动态规划过程如下：

$$EXACT\left(j, \begin{bmatrix} k'_1 \\ k'_2 \\ \cdot \\ \cdot \\ \cdot \\ k'_m \end{bmatrix}\right) = max_{i\in(1,m)} \left\{ EXACT\left(j-1, \begin{bmatrix} k'_1 \\ \cdot \\ \cdot \\ k'_i - 1 \\ \cdot \\ k'_m \end{bmatrix}\right) + \mathcal{R}_i(u) \right\}$$

H4 RevMax-S

通过贪心算法来求解：

1. 对每个竞争者按照 $\sum_{u\in V} A_{iu}$ 从大到小排序。
2. 根据排序，主机通过仅考虑一个竞争者来取得收入最大化，对每个竞争者 C_i 分配 $top - k$ 种子节点。
3. 为了使得种子集合不相交，每次选完都从图中删除掉这些已选的种子节点。

Algorithm 2 RevMax-Separate: Greedy Seed Set Selection

Require: Graph $G = (V, E, P)$, rev. matrix (A_{iu}), m campaigners
Ensure: Seed sets $S_1, \ldots, S_m$, each of size k

```
Sort and process campaigners in descending order of ∑_{u∈V} A_iu
for each campaigner C_i; 1 ≤ i ≤ m
for i = 1 to m do
    S_i = φ
    for j = 1 to k do
        v = arg max_{v∈V\S_i} F_i(S_i ⋃ {v})
        S_i = S_i ⋃ {v}
    end for
    V = V \ S_i
end for
Output S_1, ..., S_m
```

时间复杂度：$O(mkn(n+e)t)$

H2 数据集

Dataset	# Nodes	# Edges	Edge Prob: Mean, SD, Quartiles
Flickr	78 322	20 343 018	0.09 ± 0.06, $\{0.06, 0.07, 0.09\}$
DBLP	684 911	4 569 982	0.08 ± 0.07, $\{0.05, 0.05, 0.10\}$
NetHEPT	15 229	62 752	0.28 ± 0.28, $\{0.0006, 0.27, 0.53\}$

H2 实验

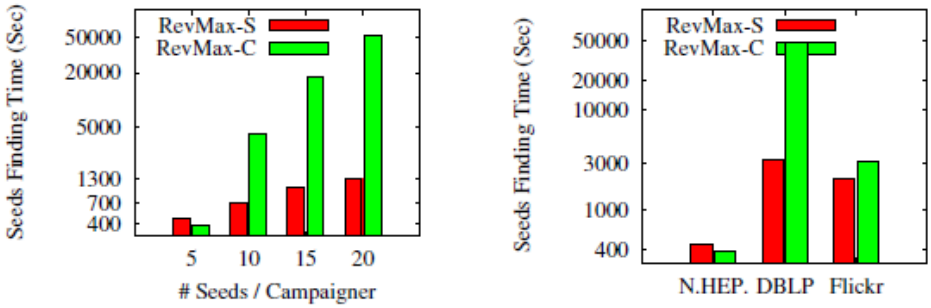
1. setting
   - 对于K-LT模型，节点入边概率和不超过1，通过入边概率和进行规范化。在所有的实验中限制Monte-Carlo采样数量为1000。
   - 设置竞争者数量2-10个，每个竞争者种子节点数为5-100个
   - 考虑三类收入分布：Uniform，Clustering with Low Competition和Clustering with High Competition。U设置每个C选择目标用户是均匀的且独立的，设置收入矩阵元素： $A_{iu} = 1, p = 1/m; A_{iu} = 0.1, p = 1 - 1/m$ 。CLC假设每个C的目标用户组成类簇中心，并假设一些用户属于所有C的目标用户，CLC是将这部分重叠用户设置一个很小的比率，首先将图分割为15个非重叠强连通类簇，前5类，设置 $A_{iu} = 1$ ，即33%属于所有C的目标用户，剩下的通过循环设置，如果一个类被设置为 C_j 的目标集， $A_{ju} = 0.5$ ，否则 $A_{iu} = 0.1, i \neq j$ 。CHC与CLC相似，将共同目标集比率设置为66%。
   - 比较方法设置为Random，RevMax-C和RevMax-S。Random跑10轮，选择最好的结果。
2. Performance: Effectiveness & Efficiency

TABLE II: Revenue Improvement Rate (RIR), MCIC Model with 2 Campaigners, *NetHEPT* Dataset

Revenue Distribution	#Seed Nodes per Camp.	RIR RevMax-S	RIR RevMax-C
CRH	5	2.52	**3.14**
	10	2.92	**3.28**
	15	2.68	**3.07**
	20	1.94	**2.23**
CRL	5	3.30	**3.33**
	10	2.91	**3.20**
	15	2.48	**2.94**
	20	2.09	**2.37**
U	5	3.23	**3.52**
	10	**2.12**	2.04
	15	2.72	**2.80**
	20	2.34	**2.52**

TABLE III: Revenue Improvement Rate (RIR), MCIC Model with 2 Campaigners, 5 Seeds/Campaigner

Dataset	Revenue Distribution	RIR RevMax-S	RIR RevMax-C
NetHEPT	CRH	2.52	**3.14**
	CHL	3.30	**3.33**
	U	3.23	**3.52**
DBLP	CRH	1.04	**1.04**
	CRL	1.02	**1.03**
	U	1.03	**1.03**
Flickr	CRH	1.43	**1.66**
	CHL	**1.20**	1.15
	U	1.01	**1.11**



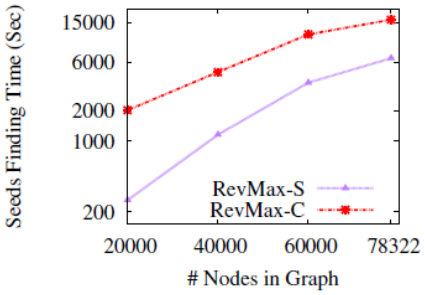
(a) *NetHEPT* dataset (b) 5 Seeds/Campaigner

Fig. 8: Seed sets finding time, MCIC model, 2 campaigners

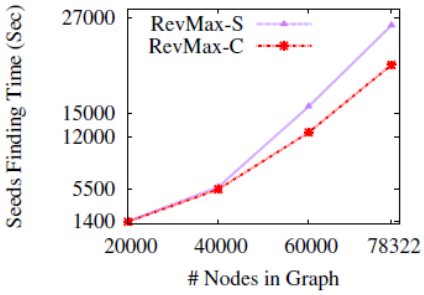
在MICI下：表2和3表明RevMax-C几乎一直比RevMax-S表现好，图8表示两种方法的效率，在小数据集NetHEPT，每个C有5个种子节点的情况下，RM-C比RM-S用更少的时间内确定to-k种子节点。而随着数据集增大，种子节点数增多RM-C需要的时间显著增加，因为它的时间复杂度为$O(ndm^2k^{2m})$，n和k的增加会使其时间高比率增加。

在K-LT下：两个方法都用了CELF++，表现情况与MCIC下差不多，m和k小的时候RM-C表现更好，m和k很大时RM-S更好。

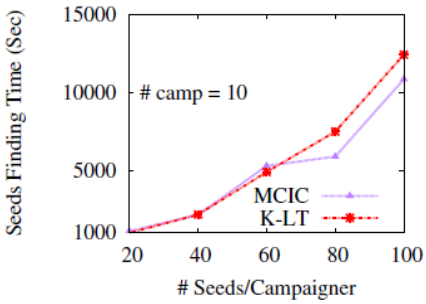
3. Scalability



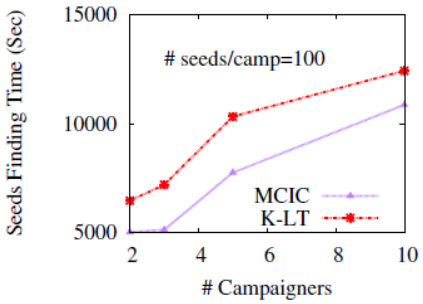
(a) MCIC model



(b) K-LT model



(a) Varying #seeds



(b) Varying #campaigners

图11分析了随着图大小增加两种方式运行时间，RM-C在MICI下为对数线性增加，在K-LT下为线性增加。

图12展示了RM-S的适应性，图结构表明RM-S随着C数量增加和S/C数量增加运行时间表现为线性。

H2 其它解决方法

Lu等人首先从社交网络主机观点方面考虑，主机代替客户端公司选择种子集合，使得每个客户端公司获得的期望营销传播接近相同。但实际上，最大化主机整体期望收入是更让人感兴趣的。

其次，他们没有考虑每个客户端公司拥有目标用户的概念，而实际上每个竞争公司都有自己的目标用户。

H2 创新之处

- 对于问题上的创新：提出了为了多方公司（竞争者）提供关键用户选择的社交网络主机收益最大化的新问题。
- 对于问题解决方法的创新：针对提出的问题提供了一个基于动态规划的近似算法和一个有效且适应性强的基于贪心策略的解决算法。