IMP

赵耀

ISE-One IC Sample

初始化ActivitySet为SeedSet
count = ActivitySet.length
while (!ActivitySet.IsAmpty())
    newActivitySet 初始化为空
    for each seed in ActivitySet
        for each inactive neighbor in seed
            seed 以自己和邻居间的权重为概率尝试去激活邻居
            if (激活成功)
                更新邻居状态　设置了一个list，只有激活的点才会被放进去
                newActivitySet.add(neighbor)
            endif
        end for
    end for
     count = count + newActivitySet.length
     ActivitySet = newActivitySet
  end while
return count

ISE-One LT Sample

```
初始化ActivitySet为SeedSet
初始化每个结点的阈值（随机产生），如果产生0.0的阈值，加入ActivitySet中
count = ActivitySet.length
while (!ActivitySet.IsAmpty())
        newActivitySet 初始化为空
        for each  seed in ActivitySet
                for each inactive neighbor in seed
                        计算该邻居结点的所有激活状态邻居的权重总和w_total
                        if (w_total>= neighbor.thresh)
                                更新邻居状态为Active
                                newActivitySet.add(neighbor)
                        endif
                end for
        end for
         count = count + newActivitySet.length
         ActivitySet = newActivitySet
  end while
return count
```

ISE

- 重复IC或LT的采样过程N次，通常N = 10000，取影响结点数目的平均值

```
sum = 0
N = 1000000
for i to N:
    oneSample  = one_LT_Sample() or one_IC_Sample()
     sum = sum + oneSample
return sum / N
```

IMP

- 通常有2类算法
- 贪心算法：爬山贪婪算法、CELF、CELF++、NewGreedy、MixGreedy、SCG等等
- 启发式算法：Degree、DegreeDiscount、PMIA等等

Greedy算法

- 每次选取边际效益最大的结点
- 每次选取都要对所有结点计算边际效益，复杂度高

CELF

- 利用子模性减少了大量的冗余计算量

启发式算法

- 不需要计算边际效益，即不进行准确评估，仅计算出某些特性，根据这些特性去优先选择结点
- 计算量小，但效果没有贪心算法好

综合使用启发式算法和贪心算法

- 先利用启发式算法选出一些候选结点
- 再利用贪心算法从候选结点中挑选