2023算法设计与分析课程报告

题目：Reachability-Driven Influence Maximization in Time-dependent Road-social Networks

姓名：陈俊乐

学号：2021113694

# 论文介绍

【介绍所阅读论文】

## 论文背景

影响力最大化作为通过社交网络加强营销和促进活动的一种手段，引起了人们的极大关注。这种特定的影响力扩散模型通常是从亲戚、朋友和同事等社会关系中构建的。这些关系之间的强大信任使信息传播比传统的基于广播的传播（如报纸）更有效。然而，随着实时位置支持服务的发展，在计算影响的传播时，自然倾向于增加对用户的时间依赖性位置信息的考虑，而不是平等地考虑所有用户。这种与时间相关的位置信息将物理世界与虚拟世界连接起来，形成了与时间有关的道路社交网络。

基于时间的道路社交网络应用程序在现实中非常有用：

（1）基于位置的推荐，它为用户提供了一个可以快速到达的最喜欢的餐厅列表；

（2） 基于位置的广告， 企业为能够在特定截止日期前到达的用户发布广告；

（3） 基于位置的紧急消息分发，官员向能够处理当地紧急情况的用户发送通知。

这些基于位置的应用不仅需要考虑影响扩散期间的位置信息，还需要考虑与时间相关的信息，因为道路网络通常是与时间相关而不是静态的。直观地说，在现实世界中，能够在截止日期前到达促销地点的用户更有可能参与该促销活动，而不是在另一个城市或过期的促销活动。这些要求需要考虑用户与时间相关的位置信息。

## 算法介绍

【设计思想】CTS索引分为三层：时间索引、区域索引和社区索引。其中，时间索引是根据给定的时间间隔（例如10分钟）划分时间，由一段一段时间组成的，可以控制TS-index大小；区域索引存储关于在时间间隔中每个节点的最大和最小可达区域的信息；社区索引基于社交网络划分为多个社区，希望利用检测到的社区的影响来近似整个网络中节点的影响，将社区内节点的影响可以近似于整个网络。具体而言，目标是确保同一社区内的节点具有较大的相互影响，而不同社区之间的节点具有较小的相互影响。

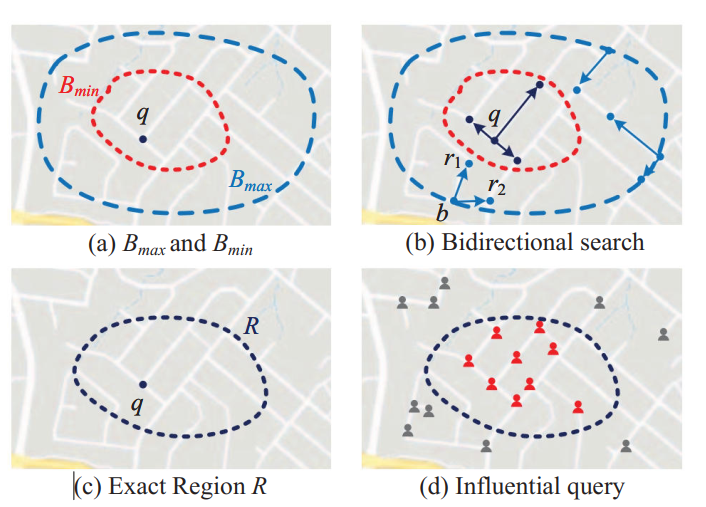
【算法执行流程】**（（1），（2）是不考虑社区的算法，为TS索引，使用（1），（2）确定满足条件的节点，然后用贪心算法求出影响力最大化的k个节点，从（3）开始考虑社区，因为（1），（2）需要遍历整个图，代价大）**

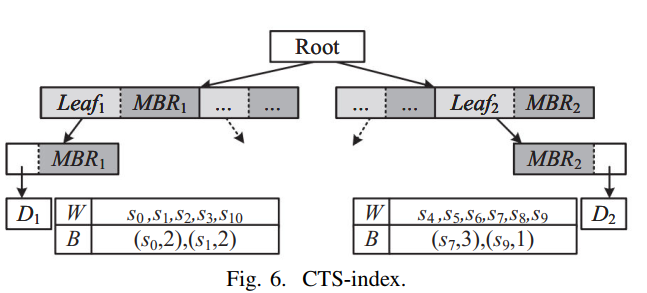
1. 最大和最小区域搜索。

最大有界区域（MaxR）搜索的主要思想是通过上界区域索引U（Gr）快速获得RDIM搜索可能达到的最大区域。该算法本质上是一个以q查询点的扩展算法。具体来说，我们使用开始时间来确定第一次访问的时间索引。然后，我们利用U（q）来获取在to所属的时间间隔内从q可达的最远节点的集合*M*1(*q, θ*) = *M*(*q, to*)，接着以t0+Δt为起始时间计算M1中的节点的可达的节点集，然后获得在下一个Δt可达的最远节点集，表示为*M*2(*q, θ*) = *∪v∈M* 1(*q,to*)*M*(*v, to* + Δ*t*)。我们一直搜索，直到第m步满足时间阈值，即(*m -* 1)Δ*t < θ ≤ m*Δ*t*。最后，q在时间上可以达到的最大边界区域Bmax是*Mm*(*q, θ*) = *∪v∈M* (*m-*1)(*q,to*)*M*(*v, to* + (*m -* 1)Δ*t*)。类似地，可以使用L（Gr）来实现最小有界区域（MinR）搜索，并获得最小边界区域Bmin。

1. 双向搜索

双向搜索算法的主要思想是通过向前和向后扩展搜索，快速计算q在截止日期之前可以到达最大和最小区域之间的哪些节点。特别地，前向搜索是从q对*L*(*Gr*)执行的。相反，后向扩展搜索是以边界节点v∈Bmax为起始节点的MaxR和MinR搜索。首先，我们在前向搜索中遍历L（q），使用L（q）中的最小距离函数从q快速获得第一个时间间隔Δt，表示为*L*(*q,* Δ*t*) = *{*(*v*1*, to* + *ω∗*(*q, v*1*, to*))*, · · · ,* (*vn, to* + *ω∗*(*q, vn, to*))*}。*对于每个边界节点v∈Bmax，我们通过向后搜索来计算其最大和最小区域，以获得在时间间隔[0，θ−Δt]内可以到达的所有节点，分别表示为MaxR（v，θ−mΔt）和MinR（v、θ−m△t）。然后，我们通过检查在截止日期θ之前L（q，Δt）和MinR（v，θ−mΔt）（分别为MaxR（v、θ−m△t））是否存在交集来检查q和v是否可达。在检查过程中，我们不断提取v的邻居节点（neighbor（v）−Bmin）∈Bmax，以供进一步研究。





CTS索引（W代表社区用户，B代表用户及其影响力）

1. 社区检测

社区检测算法本质上是一种迭代方法，通过使用通用的影响扩散模型来实现。首先，为社交网络中的每个节点v分配一个从1到N的唯一标签（N是社交网络中节点的数量）作为v.C，该标签标识该节点所属的社区。对于每个节点v，使用VI模型计算受其影响的相邻节点Neighbor（v），并将社区标签传递给这些相邻节点。迭代地，更新每个节点的社区标签，并且在每次迭代中，每个节点的团体标签由邻居（v）中的大多数社区标签确定。我们预计节点在其社区中的影响程度与其在整个网络中的影响度之间的差异很小。为了获得一组好的社区来近似整个图中一个节点的影响，我们定义了一个社区比率来衡量两个社区之间的连通性，并且当社区之间的社区比率大于阈值时，我们将两个社区合并。Cn和Cm之间存在的群落比率定义为：ε=ICm（v）\ICm（u），其中ICm（v）是v在Cm中的影响程度。近似比率成立，因为u的群落是由受u影响的节点决定的。如果ICm（ve）足够大，那么v属于Cm，u和v在同一群落中；相反，如果u和v分别属于Cm和Cn，则ICm（v）必须很小，小于u在Cm中的影响程度ICm（u）

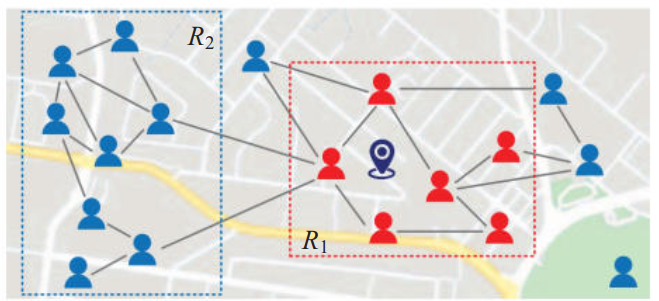
1. 基于社区的影响力最大化查询

该算法的主要思想是检查与可达区域R相交的社区，然后从这些社区中选择有影响力的节点，最后将其聚合，得到前k个节点作为最大有影响力的种子节点。

我们首先按照MBR大小的降序对M′中的节点进行排序，同时以相同的顺序更新M中的节点，因为M中的结点映射到M′中。然后构造候选种子节点集S′，以保持对应于I的Leaf中具有最大影响力的前k个用户节点。注意，通常我们选择前k个有影响力的节点作为候选种子，但如果内部节点中有影响力的总节点数小于k，则我们提取内部节点中所有有影响力的结点作为候选种子。创建一个包含Mi中种子节点的模板节点集。接下来，它进入CTS算法的核心过程，即用从M中选择的有影响力的节点更新S′。为了简洁，该过程选择M中的有影响力节点，并将其与S′中的节点进行比较，之后较大的将节点添加到最终结果集S。该过程继续进行，直到找到前k个种子节点为止。具体而言，CTS算法首先选择s′中第一个对象s的I（s），以与M′中第一对象tempM′的MBR大小进行比较。如果I（s）大于tempM′.MBR的大小，则在tempM′中没有节点u∈M的影响程度大于我（s）。更准确地说，即使u可以影响tempM′中的所有节点，I（s）也大于I（u，tempM′）。保持s不变。在这种情况下，s是s′中的第一个对象。如果I（s）小于tempM′.MBR的大小，则CTS算法需要计算tempM中最具影响力的节点u的影响程度。如果I（u，tempM′）大于I（s），则u被分配给s并插入到StempM中。然后，s被插入到结果集s中。返回包含前k个种子节点的结果集S

【例子】

假设发生了紧急情况，比如一个失踪的孩子，官员们需要组织一场搜救行动。组织者希望通过微信等社交网络传播消息，并选择有影响力的用户，以确保尽可能多的附近用户得到通知。官员们要求潜在的志愿者，即R1中的用户，能够在给定的最后期限之前到达配置地点，安排救援。这里可以通过最大最小区域和双向搜索来实现，即找到R1对应的用户，然后在R1中通过社区检测的方式聚合，找到top-k最大影响力的节点。



## 理论和实验结论

在本文中，定义了一种实用的查询类型，即RDIM查询，以找到对能够在截止日期前到达给定位置的目标用户具有最大影响的前k个种子节点。在基于时间相关道路信息和位置相关社会关系的推荐系统中，以前没有解决方案解决过此类场景，本文证明了这个问题的重要性。此外，基于时间依赖的道路社交网络的特征，提出了一个包含时间、区域和社区三个维度的索引，极大地提高了查询性能。在真实网络上的实验结果表明，该方法具有较高的精度和效率。未来工作的潜在研究方向包括以分布式方式存储和处理与时间相关的道路社交网络。

# 领域综述

【综述该领域相关论文】

## 影响力最大化领域背景

影响力最大化是指在社交网络中，寻找一组初始节点，使得以这些节点为起点，通过信息传播模型，最终能够影响到的节点数量最大。这个问题有重要的理论价值和实际价值，比如在网络营销、舆情控制、病毒防控等方面都有应用。

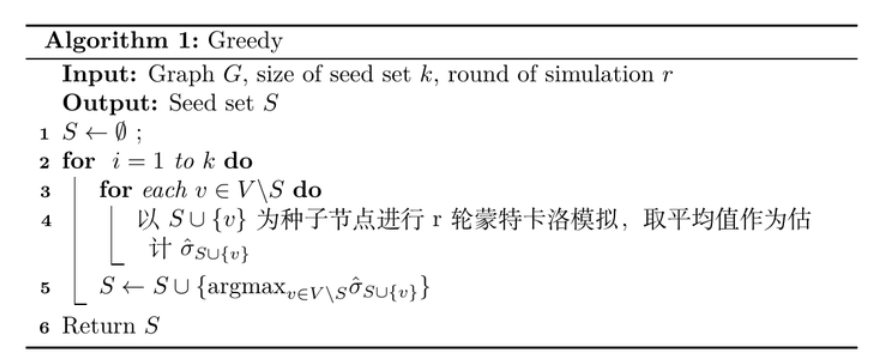
影响力最大化问题的开创性工作是Kempe等人在2003年提出的，他们定义了两种基本的信息传播模型：独立级联模型（IC）和线性阈值模型（LT），并证明了在这些模型下，贪心算法可以达到1-1/e的近似比。后续的研究主要围绕着改进算法的效率、精度和可扩展性展开。

影响力最大化问题被证明是NP-hard的，所以目前没有多项式时间的精确算法。现有的算法主要分为两类：贪心算法和启发式算法。贪心算法基于局部最优选择来近似全局最优解，精度相对较高，但是运行时间较长，需要大量的蒙特卡罗模拟来估计影响力。启发式算法基于一些启发策略来选择初始节点，不需要精确计算影响力，运行时间较短，效率较高，但是精度较差。

## 有(1-1/e)近似比的贪心算法方法介绍

Kempeetal.提出了第一个有理论保证的近似算法，算法是(1-1/e-ϵ)-近似的，即保证最后得到的种子集合S满足σ(S)≥(1-1/e-ϵ)⋅OPT，其中OPT=(max)(|S^' |≤k) σ(S')为影响力最大值。在次模函数定理(Nemhauser)：对于单调次模的非负函数f，设S\_k 是通过每一步选择一个边际增益最大的点，共执行k步得到的大小为k的集合。（即设S\_0=∅，第i步选择点v=argmaxf(S\_(i-1) ∪{v})-f(S\_i-1)，并添加点v得到S\_i=S\_(i-1)∪{v})。

由以上定理，很快就能得到贪心算法的流程。在函数是非负单调次模的模型下，直接按照定理内容贪心就能得到（1-1/e）近似的结果了。每次都采用蒙特卡洛方法对固定的进行估计。



## 考虑商品（物品）之间的关系（竞争，互补等）的影响力最大化（效用驱动）类方法介绍

将能够产生最高正效用的物品（即互补）捆绑在一起进行分配。基于项目预算约束，这样的捆绑包的预算受捆绑包中项目的最小预算限制。首先初始化为物品集合，第一次清空。我们迭代地寻找整个物品集中具有最大正效用的子集(记为C)，并将其插入到物品绑定集 中。

## 影响力最大化问题的转化方法介绍

从直观上看，对从图中随机选取的点u，如果点v的影响力较大，那么v就比其他点更可能影响u。所以只要从图中随机选择足够数量的u并求出其反向可达集(设v是图G中一个给定的节点，g是给定的按上述方法生成的子图。v在g中的反向可达集g中可到达v的节点集合)，那么在这些可达集中出现频率较高的点就很可能是影响力较大的点。同样的，对于集合S来说，如果S的影响力较大，那么它能影响到的u的数量一般也越多。所以我们的问题变成了求一个至多k个点的集合S，使得S能到达的u最多，也就是S能cover的rr set最多。

# 改进点概述

【描述改论文可以改进的点】

文中定义了一个社区比率（节点在社区的影响的节点数）来衡量两个社区之间的连通性，并且当社区之间的社区比率大于阈值时，我们将两个社区合并，通过这种方法相当于把节点聚类。

然而只考虑社区比率，可能会忽略掉一些影响力小，但是能发挥重要作用的人，而影响力大的人，他可能只能影响的人，都是那些没有什么作用的人，因此不能单纯地用影响力来衡量，可以考虑加权，即如果影响到的人作用大，权值高，影响到的人作用小，权值低，形成一个加权的影响力计算函数。

# 算法与分析

## 算法设计思想

只考虑社区比率，可能会忽略掉一些影响力小，但是能发挥重要作用的人，而影响力大的人，他可能只能影响那些没有什么作用的人，因此不能单纯地用影响力来衡量，可以考虑加权，即如果影响到的人作用大，权值高，影响到的人作用小，权值低，形成一个加权的影响力计算函数。

在这里把权值大的节点拆成多个节点，但是在同一个位置，举个例子，如果一个节点t的权值为3，它在位置p，那么在位置p上就有三个节点，然后建立CTS索引，CTS叶节点的MBR对应的W集合中的t变成t1,t2,t3，B集合中，如果原来有（S0，2）,且S0能够影响t，那么就变成(S0,4)。

## 算法过程描述

【自然语言描述】按照MBR大小对节点集合进行降序排序。然后从中I（满足在要求区域里面的MBR的节点集合）中中选择top-k个最有影响力的节点S’，然后与M集合（部分与要求区域相交）的节点集合进行比较，若S’中的节点s的影响力I(s)>M集合的MBR,那么s节点的影响力肯定大于M集合的MBR中所有节点集，那么肯定可以将s节点纳入结果集合里面，否则则继续判断，直接计算M集合的MBR节点集的影响力然后比较影响力。

【伪代码叙述】

**Input:** *Q* = (*k, q, t, θ*), *I*, *M* and *M′*

**Output:** *S*: Top-*k* influential nodes

**1** Sort the nodes in *M′* in descending order by MBR;

**2** Update *M* according to the order of the nodes in *M′*;

**3** Create candidate seed nodes sets *S′*;

**4** Insert the top *k* nodes in *I.B* into *S′* in descending

order by influenced degree;

**5** Create a set *SMi* holding the seed nodes in *Mi*;

**6 while** *S.size*() *< k* **do**

**7** *s ← S′.pop*();

**8** *tempM ← M.pop*(); *tempM′ ← M′.pop*();

**9 if** *I*(*s*) *< tempM′.MBR* **then**

|  |  |
| --- | --- |
| **10** | Select*u* = *arg maxv∈tempM\StempM*(*I*(*StempM∪* |

*{v}, tempM′*) *- I*(*StempM, tempM′*));

**11 if** *I*(*s*) *< I*(*u, tempM′*) **then**

**12** *s ← u*;

**13** *StempM* = *StempM ∪ u*;

**14** *S* = *S∪s*;

**15** Return *S*;

【伪代码逐行解释】我们首先按照MBR大小的降序对M′中的节点进行排序，同时以相同的顺序更新M中的节点，因为M中的结点映射到M′中（第1-2行）。然后构造候选种子节点集S′，以保持对应于I的Leaf中具有最大影响力的前k个用户节点（第3-4行）。通常我们选择前k个有影响力的节点作为候选种子，但如果内部节点中有影响力的总节点数小于k，则我们提取内部节点中所有有影响力的结点作为候选种子。创建一个包含Mi中种子节点的节点集（第5行）。接下来，用从M中选择的有影响力的节点更新S′。该过程选择M中的有影响力节点，并将其与S′中的节点进行比较，之后较大的将节点添加到最终结果集S。该过程继续进行，直到找到前k个种子节点为止（第6-14行）。

【例子说明】

假设发生了紧急情况，比如一个失踪的孩子，官员们需要组织一场搜救行动。组织者希望通过微信等社交网络传播消息，并选择有影响力的用户，以确保尽可能多的附近用户得到通知。**同时想要尽可能地找到合适的人选（如体力好的人，消防员职业等等），这样的人能够提高救援效率**，因此，如果有一个人，他认识很多体力好的人，那么他的影响力肯定会比那些只是单纯认识瘦弱的人（认识人的数量相同）的影响力要好，选择认识更多体力好的人的人，这样子可以大大提高救援效率。

## 算法的结果

【需要和第3节所述改进点对应，论述为何解决改进点提出的问题】

原文没有考虑影响到的人的特性，在这里我简单定义为影响力效率，它是由影响力的人乘以对应权重，原文中相当于把所有权重都设置为1，没有考虑影响的人选，这样子可能会忽略到一类人（比如在搜救中，有一个人，他只认识一位朋友，但是这位朋友是消防员，消防员对搜救活动的效率提升的意义是很大的），而我提出的算法，将权重大的节点拆成多个小节点，既可以利用原文的算法和索引进行计算，还可以大大提升救援效率。