Y3226524

中国科学技术大学

University of Science and Technology of China

硕士学位论文

论文题目 社交活动网络中的影响力最大化方法研究

作者姓名 赵鹏鹏

学科专业 计算机软件与理论

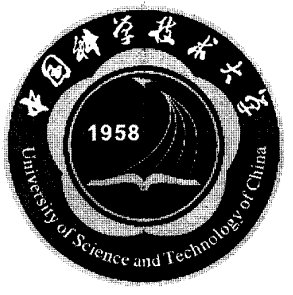
导师姓名 李永坤 副教授

完成时间 二〇一七年四月

万方数据

中国科学技术大学

硕士学位论文



社交活动网络中的影响力最大化方法研究

作者姓名：赵鹏鹏

学科专业：计算机软件与理论

导师姓名：李永坤 副教授

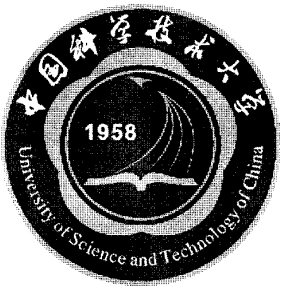
完成时间：二〇一七年四月十九日

万方数据

Y3226524

University of Science and Technology of China

A dissertation for master's degree



Research on Influence Maximization in Social Activity Networks

Author : Pengpeng Zhao

Speciality : Computer Software and Theory

Supervisor : Associate Prof. Yongkun Li

Finished Time : April 19th, 2017

万方数据

中国科学技术大学学位论文原创性声明

本人声明所呈交的学位论文，是本人在导师指导下进行研究工作所取得的成果。除已特别加以标注和致谢的地方外，论文中不包含任何他人已经发表或撰写过的研究成果。与我一同工作的同志对本研究所做的贡献均已在论文中作了明确的说明。

作者签名：赵鹏鹏　　签字日期：2017.5.24

中国科学技术大学学位论文授权使用声明

作为申请学位的条件之一，学位论文著作权拥有者授权中国科学技术大学拥有学位论文的部分使用权，即：学校有权按有关规定向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅，可以将学位论文编入《中国学位论文全文数据库》等有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。本人提交的电子文档的内容和纸质论文的内容相一致。

保密的学位论文在解密后也遵守此规定。

☑公开　☐保密 _____ 年

作者签名：赵鹏鹏　　导师签名：李永坤

签字日期：2017.5.24　　签字日期：2017.5.24

万方数据

摘 要

随着社交网络的流行，找到一组最具影响力的用户（或节点）以便触发最大的影响范围是很有意义的。例如，公司可以通过提供免费样品/折扣给这些影响力最大的用户，从而借助“口碑”效应来触发更大范围的宣传和购买，最终使该产品在网络中得以大规模普及。这类任务通常被建模为影响力最大化问题，并在过去十多年得到了广泛的研究。但是，考虑到社交网络中的用户可以参加各种各样的在线活动，例如，给产品做评价，加入讨论组等，因此影响力通过在线活动传播的情况变得更加有研究意义。

本文主要研究将用户活动考虑在内的影响力最大化问题。包含用户活动的社交网络可以被称为社交活动网络，本文首先将传统的影响力传播模型扩展至其中，从而引出社交活动网络中的影响力最大化问题。基于随机游走理论，本文定义了影响力中心性指标来近似节点集合的影响力，并提出了基于蒙特卡洛的近似算法，实现了对影响力中心性的快速计算。根据子模函数的性质，本文进一步提出了贪心算法和若干优化技术，实现了对影响力最大化问题的快速求解。最后我们利用实际的社交评分数据验证了算法的有效性、效率和通用性。可以看到本文所提出的算法在运行速度上明显优于当前已有的最优算法，且在不同的影响力传播模型与异构网络中都有很好的效果。

关键词： 社交网络，影响力最大化，随机游走

万方数据

ABSTRACT

With the popularity of online social networks, finding a set of most influential users (or nodes) so as to trigger the largest influence cascade is of significance. For example, companies may take advantage of the "word-of-mouth" effect to trigger a large cascade of purchases by offering free samples/discounts to those most influential users. This task is usually modeled as an influence maximization problem, and it has been widely studied in the past decade. However, considering that users in OSNs may participate in various kinds of online activities, e.g., giving ratings to products, joining discussion groups, etc., influence diffusion through online activities becomes even more significant.

In this thesis, we study the impact of online activities by formulating the influence maximization problem for social-activity networks (SANs) containing both users and online activities. To address the computation challenge, we define an influence centrality via random walks to measure influence, then use the Monte Carlo framework to efficiently estimate the centrality in SANs. Furthermore, we develop a greedy-based algorithm with two novel optimization techniques to find the most influential users. By conducting extensive experiments with real-world datasets, we show our approach is effective and efficient when we needs to handle large amount of online activities. Moreover, it can be generalized and scaled.

Keywords: Online Social Networks, Influence Maximization, Random Walk

万方数据

目 录

万方数据

万方数据

插图索引

万方数据

第一章 绪论

本章摘要： 社交网络的大规模流行为使用影响力来实施病毒营销，信息推广等应用提供了有效的平台。影响力的研究在社会科学和市场学等领域存在已久，计算机领域对影响力的研究伴随着大数据，数据挖掘兴起的时代背景，其涵盖了随机模型，优化算法，博弈论等理论知识。本章先从四个方面来介绍研究背景，其分别是影响力传播研究的应用，影响力传播模型，传播模型参数学习和影响力最大化。然后介绍本文的主要研究工作和贡献，即如何在考虑用户活动的情况下，解决影响力最大化问题。在本章最后，我们会给出本文的组织结构。

1.1 研究背景

自 1971 年人类的第一封电子邮件诞生起，互联网的发展就一直伴随着社交元素，特别是近十年来，社交网络的用户规模更是呈现出爆炸式的增长，以提供社交功能为核心的科技公司更是成为了创新，开放的代名词。根据最新的统计，截至 2016 年 12 月 31 日，脸书的月活跃用户数超过了 18.6 亿，同比增长 17% [1]，而国内的新浪微博月活跃用户数也达到了 3.13 亿，全年新增活跃用户 7700 万 [2]。

1.1.1 影响力传播研究的应用

人类的社会性决定了一个人对其周围其他个体和群体的依赖关系。小到听一首歌曲，看一场电影，大到对国家政策和国际政治的看法，人们的决定往往会受到周围环境的影响。而互联网的出现更是极大地丰富了人们交流和分享信息的方式，基于互联网，特别是社交网络，影响力的研究可以有各种各样的应用场景，比如病毒营销，消息推广等。

1.1.1.1 病毒营销

病毒营销靠发掘用户在社交网络中的影响力，然后借助激励措施使少数重要用户主动参与到产品的推广过程中，最终基于口碑效应来达到产品推广的目的。病毒营销的关键是确定网络中影响力最大的用户。

口碑效应是指信息通过人与人之间的交流来传播的现象。非互联网时代，人与人之间的交流主要是口头交流，民俗和神话故事的代代相传就是口碑效应的一种体现。互联网浪潮下，口碑效应有着更为广泛的应用，人与人之间的交流不仅仅局限在口头上，网络本身，特别是社交网络成为了人们分享见闻，传

万方数据

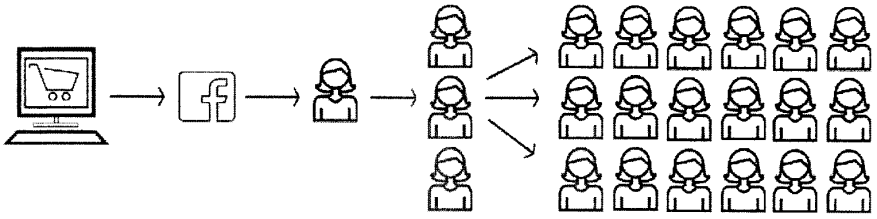


图 1.1　病毒营销示例

播信息的强大渠道。社交网络中的用户可以在网络上分享对一些产品的评价，对某一件事情的看法等，当这些信息被网络中的其他用户捕捉后，他们就可能主动在网络中继续传播该信息，如此进行下去，信息就可能在网络中大范围普及。将口碑效应用于市场推广时，我们只需对少部分影响力很大的用户采取激励措施，信息源和口碑效应的同时存在会产生级联效应，最终使得要推广的产品在网络中普及。病毒营销的成功案例很多，比如 2011 年上映的电影《失恋 33 天》，借助微博和视频网站的前期推广，成为了中国最卖座的小成本电影之一 [3]；再比如 2014 年网络上流行的冰桶挑战也是借助病毒营销策略在极短的时间内为各地支持 ALS 患者的慈善机构筹得了大量捐款 [4]。

1.1.1.2　信息传播

社交网络有着庞大的用户群体和高效的信息传播渠道，因此社交网络是消息发布和传播的重要平台。本文对人民日报两会期间发布的微博进行大致统计，得到了以下数据：

发布微博总数	转发总数	评论总数	点赞总数	单条最大转发数	单条最大评论数	单条最大点赞数
223	439,089	179,923	860,712	35,042	20,180	94,116

表 1.1　3 月 3 日至 16 日人民日报发布的有关两会的微博统计情况

从表1.1我们可以看出两会期间人民日报发布的相关微博数量在 223 条左右，平均每天超过 15 条，从该发布频率可以看出借助微博来传播两会信息相对于纸质媒介更具时效性。从转发总数，评论总数，和点赞总数综合来看，直接参与互动的网络用户已经达到了百万的数量级，考虑到部分网络用户虽然接收到信息但并未直接参与互动，实际覆盖人群会更多。此外，本文也统计了单条微博的最大转发量，最大评论量和最大点赞量，这些指标也都在上万的数量级上。需要强调的是该统计数据只考虑直接受影响的人群，通过转发、点赞等形成的级联效应会使得被覆盖的网络用户数大幅度增加。可以看出借助社交平台形成的影响力对于政策宣传之类的信息传播有着非常巨大的作用。

万方数据

1.1.1.3 其他应用

影响力研究在社会科学和市场学等领域有着广泛的应用。哈佛大学的研究者在 2007 年和 2008 年的研究 [5, 6] 中指出肥胖和吸烟行为会在社交网络中相互影响和传播。他们以 12,067 个参与者组成的社交网络为研究对象，并以这些参与者在 32 年间的医疗数据为依据，发现肥胖和吸烟行为在社交网络中存在着聚集效应，且这些行为的出现与其社交人群中是否有相同行为有直接联系等。这些发现证明了影响力在人群中确实存在，且可以为政府减少肥胖和吸烟行为的努力提供指导作用。

为了在更大规模上对影响力传播进行研究和应用，借助脸书、微博等大型在线社交网络成为了首选。一项基于脸书中 130 万用户的研究 [7] 表明年轻用户比老年人更容易受家人、朋友等的影响，男人比女人更有影响力等；而另一项基于 6100 万脸书用户在美国 2010 年大选期间的研究 [8] 表明，群众的政治倾向会受到周围朋友的影响，且这种政治见解相互影响的情况在关系紧密的朋友之间更为明显。这些发现充分表明影响力可以通过社交网络来传播，也为借助社交网络来推广国家政策等提供了有力指导。

1.1.2 影响力传播模型

社交网络可以用有向图 $G(V, E)$ 来表示，其中 V 对应网络中的用户集合，E 对应网络中用户关系的集合。网络中用户 u 和用户 v 之间的关系可以用一条有向边 $(u, v) \in E$ 来表示，有向边的方向对应着影响力的传播方向，比如微博中，用户 v 关注了用户 u，那么用户 u 就可以通过这种关注关系来影响用户 v。在影响力传播模型中，每条边还会有对应的权重，它代表了影响力的强弱。

网络中进行传播的对象可以是某个实实在在的产品，也可以是抽象的观念看法等。这些对象通过网络中的用户进行传播，一个用户对应着两种状态中的一个：如果用户接受了该产品或看法，那么该用户就处于接受状态；否则就处于未接受状态。用户的状态可以在两者之间进行转换，根据不同的影响力传播模型，状态转换会有不同的限制。但一个用户一定可以从未接受状态转换为接受状态，这被称为激活，激活对应着用户因为受到周围邻居的影响而购买了某件产品或者接受了某种观点。处于接受状态的用户可以继续尝试激活其出邻居节点，从而使得传播对象在网络中得以扩散。

影响力传播模型规定了传播对象在网络中是如何扩散的。它涉及传播对象扩散的源头，原本处于未接受状态的用户在何种条件会被激活转换为接受状态，影响力传播何时终止等问题。影响力传播模型分为很多种，其中多为随机模型。

目前研究的最深入广泛的随机模型是独立级联模型和线性阈值模型 [9]。这两个模型是由 Kempe 等人总结前人在社会科学和市场学等领域的模型 [10, 11] 后总结归纳出来的。本文重点介绍这两个模型，后边的内容也会围绕这两个模型展开。

万方数据

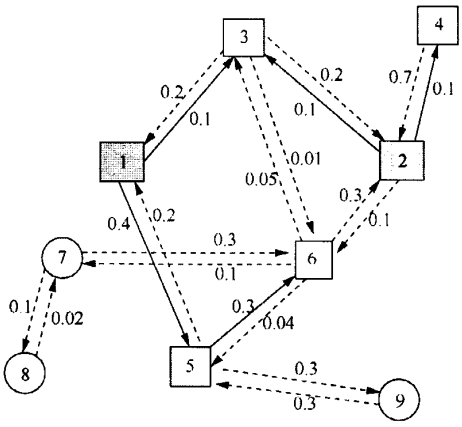


图 1.2　独立级联模型示意图

1.1.2.1　独立级联模型

对于独立级联模型来说，有向图中每条边的权重是一个概率值，我们用 $p(u,v) \in [0,1]$ 来表示，它对应节点 u 激活节点 v 的概率。令 A_i 表示在第 i 轮处于接受状态的节点集合，初始状态下 A_0 即为我们选出的种子节点集合 S。考虑第 $i \geqslant 1$ 轮，对于一个处于未接受状态的的节点 v，如果 $u \in A_{i-1}$ 并且 $(u,v) \in E$，那么 u 就会以 $p(u,v)$ 的概率激活节点 v。如果激活成功，那么 v 就变为接受状态且将节点 v 加入 A_i；如果失败，且 v 的其他入邻居也未能在第 i 轮成功激活 v，那么节点 v 在第 i 轮就保持未接受状态。假设每条有向边对应的激活概率都为 p，如果在第 i 轮，节点 v 有 r 个邻居位于 A_{i-1} 中，那么 v 在第 i 轮被激活的概率就为 $1-(1-p)^r$。当某一轮不再有新的节点被激活时，该过程停止。

图1.2是独立级联模型下一次传播过程的示意图。其中节点 1、2 为种子节点，我们用实心方框表示。我们用实线边表示激活成功，虚线边表示激活失败。在第 $i=0$ 轮，只有节点 1、2 处于接受状态，即 $A_0=\{1,2\}$；在第 $i=1$ 轮，节点 1、2 会尝试激活节点 3、4、5、6，并且成功激活节点 3、4、5，我们用空心方框表示被激活的节点；在第 $i=2$ 轮，节点 5 会尝试激活节点 6、9，并成功激活节点 6；在第 $i=3$ 轮，节点 6 尝试激活节点 7，但激活失败。由于在第 $i=3$ 轮没有新的节点被激活，因此传播过程结束。我们用圆圈表示传播过程结束后仍未被激活的节点，他们分别是节点 7、8 和 9。令 $T(S)=\cup_{i\geqslant 0}A_i$，其对应传播过程结束后所有处于接受状态的节点集合。由于有向图中的概率值引入了随机性，因此 $T(S)$ 是一个随机集合，对应传播结束后处于接受状态的节点。我们将它的期望值 $E[|T(S)|]$ 表示为 $\sigma(S)$，并称之为影响力延展度。

万方数据

1.1.2.2 线性阈值模型

线性阈值模型中，图中的每条有向边都对应着一个权重 $w(u,v) \in [0,1]$。该权重反应节点 v 在被其入邻居激活的过程中，u 所起作用的大小。对于每个节点 v，它会从 $[0,1]$ 之间随机均匀地选择一个阈值 θ_v，且其入边对应的权重满足 $\sum_{u \in N^-(v)} \leqslant 1$。同独立级联模型一样，我们令 A_i 表示在第 i 轮处于接受状态的节点集合，初始状态下 A_0 为种子节点集合 S。对于第 i 轮中处于未接受状态的节点 v，如果 $\sum_{u \in N^-(v) \cap (\cup_{0 \leqslant j < i} A_j)} w_{uv} \geqslant \theta_v$，那么 v 就会在第 i 轮被激活为接受状态，并且加入 A_i 中；否则节点 v 在第 i 轮保持未激活状态。当没有新的节点被激活时，该过程结束。由于 θ_v 是随机生成的，因此每一组 θ_v 下处于接受状态的节点集合 $T(S)$ 是不同的，同独立级联模型一样，我们将 $\sigma(S) = E[|T(S)|]$ 称为线性阈值模型下种子集合 S 的影响力延展度。

1.1.3 传播模型参数学习

影响力传播模型描述了影响力在网络中进行扩散的过程。可以看到，无论是独立级联模型中的影响概率，还是线性阈值模型中的影响权重或节点被激活的阈值，都说明了影响力传播模型中存在大量的参数需要去确定。因此要使影响力传播研究在实际中发挥作用，基于网络结构和用户行为数据来学习传播模型中的参数信息是必不可少的。

影响力传播学习的主要依据是如果存在影响关系的两个用户在一定的时间间隔内先后执行了相同的动作，且动作发生的先后顺序与影响力传播方向一致，那么就可以认为先执行这一动作的用户影响了后执行该动作的用户。对于发布微博的情况，如果在用户 A 发布微博 L_1 之后，用户 B 紧接着也发布了微博 L_1，即存在 (A, L_1, t_1) 和 (B, L_1, t_2)（t_1 和 t_2 满足 $0 < t_2 - t_1 < \delta$）这样的两条记录；同时，我们可以根据网络拓扑得知用户 B 在时刻 t 关注了用户 A，即存在记录 (A, B, t)。如果时间序列上满足 $t < t_1$，这就说明在用户 A 和 B 建立影响关系后，用户 B 在短时间内执行了与用户 A 相同的行为，因此可以认为用户 B 受到了用户 A 的一次影响。如果用户 B 经常在用户 A 之后执行相同的动作，那么我们就可以推测用户 A 可以对用户 B 产生较大的影响。

根据以上认识，我们就可以确定用户间影响的概率（权重），直观上来讲，在一个节点执行某一动作前，其入邻居中可能有多个都提前执行过该动作，此时就涉及到影响力如何分配的问题。现有方法主要分为两类：一类是最大似然估计，另一类是基于信用分配的频度分析。

1.1.3.1 最大似然估计

最大似然估计在设置影响力传播模型中的参数时，其目标是使得根据这些参数得到的传播结果与实际出现的影响力传播结果似然度最大 [12, 13]。简单来说，网络中用户执行的不同行为会非常多，比如微博中用户发布的微博数，对

万方数据

一个用户来说，其某一个动作发生前可能会有多个邻居用户已经执行过该动作，但当考虑了大量动作后，如果一个用户经常在其某一个邻居之后执行相同的动作，那就可以认为此用户有很大可能受到该邻居的影响，即这个邻居用户对它的影响力很大。最大似然估计就是对此直观想法进行量化的手段。由于影响力传播模型中涉及到大量的参数，因此直接使用最大似然估计会很难计算，所以通常会使用期望最大化算法等。但这种算法需要多次迭代，在参数很多的大图中计算效率往往很差，而且期望最大化算法不能保证得到全局最优解。

1.1.3.2 信用分配和频度分析

最大似然估计无论是分析还是计算都过于复杂，对此有人提出了基于信用分配的频度分析方法 [14]。该方法不需要保证所得参数与最终影响力传播的实际结果似然度最大，它会启发式的根据实际传播数据直接分配邻居对某一节点的影响，并将所分配的影响称为信用值，然后对每个邻居节点最终得到的信用值做频度分析处理来得到用户间的影响概率（权重）。其基本做法是，对于某一个传播对象，如果一个节点在执行该动作前已经有多个邻居节点执行了该动作，那么就按一定的规则将影响该节点的信用值分配给这几个邻居节点。简单情形下可以认为这些邻居节点会平分信用值，更复杂的情况是，每个邻居分到的信用值会和动作执行的间隔等有关系，比如间隔越小，分得的信用值越高。当对所有的传播对象都执行完信用分配后，再由直接的频度分析得到最终的影响概率（权重）。比如可以将某个节点 u 针对节点 v 得到的总信用值除以节点 u 参与过的总动作数来得到节点 u 对节点 v 的影响力大小。基于信用分配的频度分析法计算效率很高，因此可以扩展到大图的影响力传播学习中。

1.1.4 影响力最大化

研究影响力传播过程，建立影响力传播模型的主要目的是控制和优化影响力的传播。其中最重要的问题是如何选择合适的种子节点使得影响力延展度最大，即影响力最大化问题。本节简要介绍独立级联和线性阈值模型中的影响力最大化问题。第二章会详细介绍目前已有的影响力最大化算法。

影响力最大化问题的定义如下：给定社交网络对应的图结构 $G(V,E)$ 和影响力传播模型，在 V 中选取出由 k 个节点组成的种子集合 S^*，其中 k 由产品推广方所能接受的代价决定，最终使得 S^* 在此影响力传播模型下所产生的影响力延展度 $\sigma(S^*)$ 最大。即影响力最大化对应着组合优化问题 $S^* = \mathrm{argmax}_{S\subseteq V,|S|=k}\sigma(S)$。

1.1.4.1 子模函数

影响力最大化涉及组合优化问题，可以证明，如果可以解决独立级联或线性阈值模型下的影响力最大化问题，那么就可以解决图覆盖问题 [15]，因此影

万方数据

响力最大化问题在解决难度上至少和图覆盖问题一样。这也就是说准确解决影响力最大化问题是不可行的，因此需要求助于近似算法，即设法找到使影响力延展度近似最优的种子集合。近似最优解和实际最优解之间的比例被称作近似比，我们的期望是所设计的近似算法可以满足比较高的近似比，从而可以扩展到任意的网络中去。已有工作多是基于影响力延展度函数的子模性质来设计相应的贪心算法，从而实现影响力的近似最大化。

考虑一个集合函数 f，对于集合 S 和 T（$S \subseteq T \subseteq V$）以及 T 外的任意一个元素 $u \in V \backslash T$，如果 f 满足 $f(S \cup \{u\}) - f(S) \geqslant f(T \cup \{u\}) - f(T)$，那么就称 f 是一个子模函数。子模函数反映了将元素 u 加入到集合 S 带来的边际增益会随着集合 S 的增大而递减的现象。这种现象广泛存在于实际生活中，比如 A 有十套房子，B 只有一套，那么再给 A、B 一套房子所带来的幸福感提升，A 一定小于 B。可以证明，独立级联和线性阈值模型下的影响力延展度函数都具有子模性质。

经常和子模性质一起使用的是函数的非递减性，对于任意两个集合 S 和 T（$S \subseteq T \subseteq V$），如果集合函数 f 满足 $f(S) \leqslant f(T)$，就称 f 满足非递减性。同样可以证明，独立级联和线性阈值模型下的影响力延展度函数具有非递减性。

1.1.4.2 贪心算法

非递减子模函数的重要性质是可以用简单的贪心算法得到所对应优化问题的近似最优解。对于影响力最大化问题，当需要输出 k 个种子节点时，贪心算法需要运行 k 轮，每一轮找出一个节点 u 使得当前种子集合 S 的边际增益 $\Delta(u) = \sigma(S \cup \{u\}) - \sigma(S)$ 最大。可以证明，贪心算法输出的近似解满足 $1 - 1/e$ 的近似比 [16]。

简单贪心算法虽然可以在多项式时间内找出影响力最大化问题的近似最优解，但对于当今的社交网络规模来说，其时间开销仍然难以忍受，下一章会具体介绍一些优化过后的影响力最大化算法。

1.2 本文主要研究工作和贡献

考虑到当今的社交网络中，用户可以参加各种各样的在线活动，例如脸书中的用户可以加入一些讨论组，微博中的用户可以参加一些热门话题等，也就是说社交网络中的用户不仅可以创建直接关系，也可以通过参加各种在线活动形成间接关系。例如，如果两个用户共同参加了微博中的某个话题，那么用户的评论，看法等就可以通过该话题进行传播。此时无论参与该话题的用户间是否存在关注关系，他们之间都会有影响力的存在。本文将这种用户间既存在直接关系又存在通过活动形成的间接关系的网络称为社交活动网络。

考虑到社交活动网络中，影响力既可以沿着直接关系传播，又可以沿着间接关系传播，因此有必要基于这两种关系来对用户的影响力进行分析。在本文

万方数据

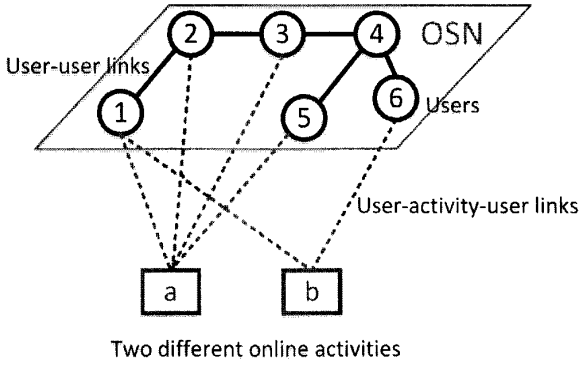


图 1.3　社交活动网络示意图

中，我们只关注对用户行为产生积极影响的在线活动，例如，在线评分系统中打分较高的评价和用户加入有共同兴趣爱好的群组等。图 1.3显示了一个简单的社交活动网络，其包含了六个用户和两个不同的在线活动。特别地，在线活动 a 和 b 可以是两种不同类型的活动。如果图1.3表示一个在线评分系统，则用户连接到同一活动意味着他们给同一产品做出了比较高的评价，例如连接（1，b）和（6，b）意味着用户 1 和用户 6 对产品 b 给出了较高评价。此时虽然用户 1 和用户 6 在网络中没有直接联系，但用户 1 仍然可能影响用户 6 在未来的购买行为。

大量在线活动的存在使得影响力通过用户活动传播的情况变得更为重要。因此单单考虑直接关系会严重影响所选种子节点集合的准确度。为了进一步说明问题，我们再次考虑图 1.3中的情形，如果只考虑直接关系，且以顶点的度数来衡量用户的影响力，此时用户 4 应该是最有影响力的节点；但将用户活动考虑在内后，由于用户 1 可以通过活动影响网络中的大部分用户，因此用户 1 会成为网络中最有影响力的节点。

现有影响力最大化算法主要考虑直接关系，这促使我们解决将用户活动考虑在内的影响力最大化问题。该问题面临着以下几个难点：首先单纯考虑直接用户关系的影响力最大化问题已被证明是 NP 难问题，将大量的用户活动考虑在内后，该问题将变得更加复杂；其次大量用户活动的存在使得底层的关系网络变得极为稠密，因此传统做法的时间开销会变得非常大。为此本文提出了新的计算框架来解决社交活动网络中的影响力最大化问题，该工作 [17] 的贡献主要有以下几点：

- 本文研究了社交网络中用户活动对影响力传播的促进作用，首先构建了超图模型对社交活动网络进行了建模，然后基于随机游走理论，定义了影响力中心性指标来近似节点集合的影响力。

- 本文提出了基于蒙特卡洛的近似算法，实现了对影响力中心性的快速计算；并进一步提出了贪心算法以及若干优化技术，实现了对影响力最大化

万方数据

问题的快速求解。

- 本文利用实际的社交评分数据验证了算法的有效性、效率和通用性。通过和当前最先进的算法 IMM[18] 进行比较，可以验证本文的算法在运行速度上明显优于 IMM，且在不同的影响力传播模型与异构网络中都有很好的效果。

1.3　本文组织结构

本文的其余章节组织如下：

第 2 章介绍了一些常见的影响力最大化方法，并介绍本文要用到的随机游走理论及其应用。第 3 章具体介绍本文的工作，先给出了社交活动网络中影响力最大化问题的定义，然后实现对该问题的快速求解。第 4 章设计实验来验证算法的有效性、效率和通用性。第 5 章总结全文，并阐述对未来工作的展望。

万方数据

此页不缺内容

万方数据

第二章　影响力最大化和随机游走相关研究

本章摘要： 研究影响力传播过程的一个主要目的是选择合适的种子节点使得影响力延展度最大，即影响力最大化。基于网络拓扑来刻画节点重要性的方法很早就已出现，但其缺乏对网络中用户行为的刻画。影响力传播模型涵盖了网络拓扑和用户行为两方面的内容，是目前研究的主流。本章首先介绍一些基于网络拓扑来刻画节点重要性的中心性指标，然后介绍基于影响力传播模型的影响力最大化算法，然后介绍本文研究需要用到的随机游走理论及其应用，最后对本文的研究意义进行总结。

2.1　基于网络拓扑的中心性指标

中心性 [19] 可以基于网络拓扑结构来衡量节点的重要性。中心性大的节点在网络中处于相对关键的位置，在社交网络中这类节点可以帮助或阻断信息的扩散，在搜索引擎中可以对网页进行排名等。

2.1.1　度中心性

度中心性是最简单的中心性，指的是与一个节点相邻的节点数。它反应的是一个节点在施加或受到影响时，直接接触到的节点数。度数越大说明这个节点与其他人的关系越紧密，该点处于网络核心地位的可能性越大，也就更能有效控制及影响网络中其他用户的活动。相反，中心度越低的节点越处于边缘地位，很少参与互动交流，对其他节点的影响很小。在有向图中，度中心性一般区分为出度和入度。比如新浪微博就可以视为有向图，对于一个用户，其粉丝数表示入度，反应一个用户的受欢迎程度，关注数则表示出度，反应一个用户在网络中的参与度。

度中心性的计算高效方便，只需对图做一次遍历即可得到所有点的度中心性。由于度中心性只考虑一个节点的邻居信息，因此即便不知道整个网络的拓扑结构也可以得到一个节点的度中心性。度中心性的缺点是由于它只考虑了网络的局部信息，所以在衡量一个节点的影响力时不够准确。在应用中，节点的度中心性一般被用来做对比试验。

2.1.2　接近中心性

接近中心性是指一个节点与图中所有其他节点的最短路径的平均长度。接近中心性既可以反映一个节点对网络中其他节点的依赖关系，例如传播消息时是否需要很多其他节点的参与；又可以反应信息从开始传播到被其他所有节点

万方数据

接收的时间开销，离其他节点越近，信息传播的就越快。接近中心性的原始定义为：

$$C(v) = \frac{|V| - 1}{\sum_{v' \in V} d(v, v')}. \tag{2.1}$$

其中，$d(v, v')$ 表示节点 v 到节点 v' 的最短路径长度。

接近中心性考虑了网络的全局信息，因此能更准确的反应节点的重要程度。缺点是由于它是基于最短路径定义的，所以它默认信息是基于最短路径传播的。并且最短路径本身的计算开销很大，使用斐波那契堆虽然可以达到 $O(m + n\log n)$ 的时间复杂度，但要选出接近中心性最大的节点集合需要计算所有点对之间的最短路径，因此时间开销会达到 $O(nm + n^2\log n)$，这在大规模的社交网络中是无法接受的。

利用接近中心性来寻找最有影响力的 k 个节点集合的工作有很多，主要分为准确计算和近似计算两类。目前准确计算最先进的算法是 2014 年发表在 ICDE 上的工作 [20]，该工作的主要思想是在计算过程中重用中间计算结果来减少最终的时间开销，同时辅以剪枝操作，排除对不可能出现在最终结果的节点的计算，最后再对节点的计算顺序进行调度和优化，从而大大减少了计算开销。该算法支持有权图的计算，并且相对于传统的准确计算方法，其计算速度提高了几十倍。但其仍难以扩展到大规模的社交网络中。

近似计算最先进的算法是 2014 年发表在 WWW 上的工作 [21]。该工作的主要思想是对接近中心性的定义进行扩展，扩展后的接近中心性是一个非递减的子模函数，因此可以设计有效的贪心算法获得有理论保证的近似最优解。在贪心算法的基础上他们又使用 FM-sketch 概率算法来快速近似计算接近中心性，从而进一步提高了计算效率。最后，该工作还设计了将大规模的图存储在磁盘上时减少磁盘 I/O 的方法，以使其算法可以进一步扩展到大图的计算中。但该算法的缺点是难以适应有权图的计算。

2.1.3　中介中心性

中介中心性描述的是一个节点位于其他所有点对之间的最短路径上的概率和。传统的中介中心性也是基于最短路径来定义的。中介中心性可以反应一个节点对网络中其他点对之间进行信息传播时的控制能力。当网络中的信息沿着最短路径传播时都需要经过某个节点，那么该节点的中介中心性就会很大。中介中心性适用于控制并阻断谣言扩散的场景，因为多数信息的流动都要经过中介中心性很大的节点，因此只要保证这些点不转发某消息，就能保证该消息不会大范围在网络中扩散。中介中心性的定义为：

$$C_b(v) = \sum_{j<k} \frac{g_{jk}(v)}{g_{jk}} \quad j, k \neq v. \tag{2.2}$$

其中，g_{jk} 表示点 j 和点 k 之间的最短路径数目，$g_{jk}(v)$ 表示的是点 j 和点 k 之间的最短路径中经过点 v 的数目。

万方数据

中介中心性的计算更加复杂，单独计算一个节点的中介中心性就需要找出所有点对之间的最短路径，因此计算单个节点的中介中心性的时间开销已经达到了 $O(nm+n^2logn)$，要寻找中介中心性最大的节点集合则更加困难。

目前对中介中心性的研究主要还是计算单个节点的中介中心性。准确计算的最先进方法出现在 2001 年，该工作在无权图上计算中介中心性的时间复杂度可以减小为 $O(nm)$，但对于有权图仍是 $O(nm+n^2logn)$。关于中介中心性的近似计算的研究比较活跃，主要分为关键点采样和路径采样两种。关键点采样 [22] 的基本思想是从图中抽取部分关键点进行单源最短路径的计算，然后利用这些最短路径信息来计算指定节点的中介中心性，由于避免了对所有点对计算最短路径，因此计算速度得以提高。路径采样 [23] 每次从图中抽取一对节点并计算这对节点之间的最短路径，在进行足够多的采样之后就可以利用这些最短路径信息对指定节点的中介中心性进行近似计算。

2.1.4 PageRank 中心性

PageRank[24] 是一种根据网页之间的超链接来计算网页重要性的技术。其直观理解是一个网页的重要性是由指向它的页面的重要程度和出度共同决定的。PageRank 背后的含义可以这样解释: 假设网络中有很多浏览者，当他们在页面 u 时会以 α 的概率随机选择一个 u 所指的某个页面作为下一步，并以 $(1-\alpha)$ 的概率随机均匀的从所有页面中选择一个作为下一步。随着过程的推进，正在浏览某个页面的浏览者比例会达到稳定，这个比例即为该页面的 PageRank 值。节点 u 的 PageRank 值可以用式 (2.3) 来计算，其中 $N^-(u)$ 表示节点 u 的入邻居集合，$N^+(v)$ 表示节点 v 的出邻居集合。

$$PR(u)=\frac{1-\alpha}{n}+\alpha\times\sum_{v\in N^-(u)}\frac{PR(v)}{|N^+(v)|} \tag{2.3}$$

可以看出一个邻居节点会将其 PageRank 值平均分配给它指向的每一个节点。为了避免没有出邻居的节点传递出 0 的情况，系统引入了常量 α，其取值一般为 0.85。

计算单个节点的 PageRank 值会递归的调用其余节点，因此计算开销很大。一般 PageRank 的计算采用迭代的方法，在为图中的每个节点分配一个不为 0 的初值后，随着迭代过程的进行，所有点的 PageRank 值会不断收敛到一个稳定值。

2.2 影响力最大化算法

中心性指标可以反映节点的影响力，但因为其忽略了用户行为数据往往不够准确。影响力最大化算法是基于影响力传播模型来计算节点的重要性，然后寻找影响力最大的节点集合。影响力传播模型及其参数的学习已经在第一章做

万方数据

过介绍，可以看出影响力传播模型的确定不仅和网络拓扑结构有关，还和用户的行为数据有关，这是同基于中心性来衡量节点重要性最大的不同。当给定影响力传播模型后，我们就可以在其上设计影响力最大化算法来选取影响力最大的节点集合，本文只讨论独立级联模型和线性阈值模型下的情况。可以被证明在这两种模型下，影响力最大化问题都是 NP 难问题 [25]。

2.2.1 贪心算法

2.2.1.1 简单贪心算法

简单贪心算法 [9] 是由 Kempe 等人将影响力传播模型引入到影响力研究后提出的最初解决方案。贪心算法的流程展示在算法 2.1中，其需要执行 k 轮，每一轮我们都会选出一个使得影响力延展度 $\sigma(S)$ 增量最大的节点。

可以证明在独立级联模型和线性阈值模型下，影响力延展度 $\sigma(S)$ 是非递减的子模函数，这就保证了贪心算法最终返回的节点集合 S 和最优的节点集合 S^* 之间满足以下关系：$\sigma(S) > (1-1/e)\sigma(S^*)$。影响力延展度 $\sigma(S)$ 的计算与给定的影响力传播模型有关。准确计算给点节点集合的影响力延展度是 #P 难问题 [26]，因此基本贪心算法使用蒙特卡洛模拟的方法来对其进行计算。在给定种子集合和影响力传播模型后，根据影响力传播模型刻画的影响力传播过程，我们可以对从种子集合开始的影响力传播进行多次模拟，并对多次模拟中被影响的节点个数取平均作为最终的影响力延展度。贪心算法简单有效，但是时间开销过大，实验 [27] 表明，在一个由 15000 个节点组成的小型网络中，使用 2000 次蒙特卡洛模拟来计算延展度，即使只需要选取 50 个种子节点，简单贪心算法也需要花费好几天的时间才能完成。

input : 图 $G(V,E)$ 和整数 k
output: 由 k 个节点组成的种子结合 S
1 $S \leftarrow \emptyset$;
2 **while** $|S| < k$ **do**
3 　$v \leftarrow \arg\max_{u \in (V-S)} \sigma(S \cup \{u\}) - \sigma(S)$;
4 　$S \leftarrow S \cup \{v\}$;
5 **end**
6 *return S*;

算法 2.1: 贪心算法

2.2.1.2 CELF 和 CELF++ 算法

简单贪心算法效率很差的原因是在迭代过程中，需要不断计算将其余节点加入到当前种子集合后的影响力延展度，而计算影响力延展度本身又特别耗时。贪心算法有效的一个关键因素是子模性质的存在，CELF 和 CELF++ 算法就是通过进一步挖掘子模性质来改善算法的性能。该类算法的基本思想是利用子模

万方数据

函数边际收益递减的性质，即对于同一个节点来说，它在当前迭代中带来的边际收益不会比上一轮迭代中带来的边际收益更大。

CELF[28] 算法会维护一张表 $< u, \Delta_u(S) >$，且表中元素按 $\Delta_u(S)$ 递减排序，其中 S 是当前种子集合，$\Delta_u(S)$ 是将节点 u 加入到种子集合 S 后带来的边际收益。实现时，我们可以使用一个最大堆 Q 来维护排序信息，在每一轮迭代中，我们仅仅对堆顶部的节点进行评估，且只有在满足一定条件的情况下，才会对堆发起重排序操作。

```
   input : 图 G(V,E) 和整数 k
   output: 由 k 个节点组成的种子结合 S
 1 S ← ∅, Q ← ∅;
 2 for u ∈ V do
 3 |   u.mg ← σ({u});
 4 |   u.round ← 0;
 5 |   将 u 加入最大堆 Q，Q 按 u.mg 排序;
 6 end
 7 while |S| < k do
 8 |   u ← Q 的顶部元素;
 9 |   if u.round == |S| then
10 |   |   S ← S ∪ {u};
11 |   |   Q 弹出其顶部元素 u;
12 |   end
13 |   else
14 |   |   u.mg ← σ(S ∪ {u}) − σ(S);
15 |   |   u.round ← |S|;
16 |   |   将 u 重新插入 Q, 并且对 Q 重排序;
17 |   end
18 end
19 return S;
```

算法 2.2: CELF 算法

CELF 算法可以显著提高贪心算法的效率，算法 2.2描述了 CELF 算法的工作流程。其中，$\sigma(S)$ 表示种子集合 S 的影响力延展度，Q 中维护着与网络中的用户相同数量的信息。Q 中的元素 u 存储着二元组 $< u.mg, u.round >$，其中 $u.mg = \sigma(S \cup \{u\}) - \sigma(S)$，u.round 表示 u.mg 最近一次被更新时所处的迭代轮数。

在第一轮迭代过程中，算法要对每个节点的边际收益进行计算，并将其逐个加入到最大堆 Q 中。在其后的每一轮迭代中，算法会先检查堆顶的元素 u，并判断其边际增益是否在当前迭代中被更新过。如果是，根据子模性质，u 在当前轮的边际收益一定小于上一轮的边际收益，我们可以得出 u 必定是在当前迭代中边际收益最大的节点，此时 u 被选为种子节点加入到种子集合中。否则，重新计算 u 的边际收益，更新 u.round 为当前轮，然后将其插入 Q 并重新排

万方数据

序。该过程对应算法 2.2中的 while 循环。可以看出，除了第一轮迭代必须要对每个节点的边际收益进行计算外，其余轮只有在个别情况下才会发生对节点边际收益的计算，因此大大提高了算法效率。实验结果显示，CELF 的运行速度可以比基本贪心算法快 700 倍左右。

CELF++[29] 在 CELF 的基础上进一步挖掘子模性质来提高算法效率。CELF++ 在数据组织上不同于 CELF 的地方是它的最大堆中维护的不是一个二元组，而是四元组 $< u.mg1, u.prev\_best, u.mg2, u.flag >$，其中 $u.mg1$ 表示将节点 u 加入到当前种子集合后带来的的边际增益；$u.prev\_best$ 代表在节点 u 之前已被检测到的能带来最大边际增益的节点；$u.mg2 = \Delta_u(S \cup \{prev\_best\})$；$u.flag$ 代表 $u.mg1$ 最近被更新时所处的迭代轮数。CELF++ 的关键思想是如果本轮位于堆顶的节点 u 对应的 $u.prev\_best$ 就是上一轮选出的种子节点，那么我们就可以直接用 $u.mg2$ 来更新 $u.mg1$, 而不必重新计算 $u.mg1$。且对于一轮迭代而言，$u.mg1$ 和 $u.mg2$ 是可以同时计算出来的，这就从一定程度上减少了计算开销。根据实验结果我们可以看出 CELF++ 可以在 CELF 的基础上将计算速度提高 17-61%。

2.2.2　启发式算法

贪心算法可以保证算法最后返回的结果在准确性上有一定的理论保证，但是计算效率往往不高。一些启发式算法虽然缺乏理论保证，但其计算速度很快，而且从实验结果看，其准确性往往也跟贪心算法几乎相同。在诸多启发式算法中，PMIA 算法和 SIMPATH 算法分别是独立级联模型和线性阈值模型下比较经典的算法。

2.2.2.1　PMIA 算法

在 PMIA 算法 [30] 中，对于一个节点 u，它会找出 u 到图中各个节点最可能的影响传播路径，最终形成一个针对节点 u 的树形影响力传播结构。PMIA 算法做了两方面的近似：从节点 u 到节点 v，其忽略了树形结构以外的传播路径，这是因为影响力沿这些路径传播的可能性很小；另外，节点 u 的影响范围被限制在一定的范围内，这是因为 u 对离其过远的节点影响力会很弱。构造节点 u 的树形影响力传播结构可以利用基于斐波那契堆的 Dijkstra 算法在 $O(m + n\log n)$ 的时间复杂度内完成，这大大减少了计算开销。与做过优化的蒙特卡洛贪心算法相比，PMIA 速度提高了 1000 倍以上，而选出的种子节点其影响力和贪心算法几乎相同。

2.2.2.2　SIMPATH 算法

针对线性阈值模型的启发式算法比较经典的是 SIMPATH 算法 [31]。它同 CELF 算法一样，只有在必要情况下才重新计算边际增益。CELF 算法中使用蒙

万方数据

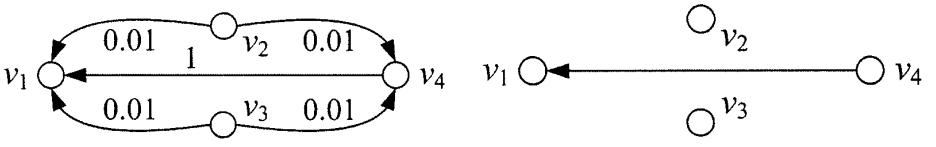
特卡洛模拟来计算边际增益会带来很大的计算开销，因此 SIMPATH 采用在节点附近区域枚举简单路径的做法来估计节点的边际增益。SIMPATH 采用了两项优化技术来进一步加快对边际增益的计算。第一个叫节点覆盖优化，这项技术使得一个节点的影响范围可以通过其邻居的影响范围直接计算出来。在第一轮迭代过程中，我们会先找一个顶点覆盖，然后使用枚举简单路径的方法计算这些顶点的影响范围，因为这是一个顶点覆盖，所以其他节点的影响范围都可以根据这些已计算点的影响范围直接得到，因此节点覆盖技术大大加快了第一轮迭代的速度。在迭代过程中，随着种子集合的增大，对影响范围的估计会变得越来越慢，因此作者又提出了向前看的优化方法来缓解这种情况。

2.2.3　反向蒙特卡洛算法

贪心算法虽然在正确性上有着理论保证，但他们的运行速度仍然过慢；而启发式算法虽然在运行时间上很快，但却缺乏理论保证。方向蒙特卡洛算法的提出首先改变了这种运行速度和理论保证不能兼顾的局面。

2.2.3.1　RIS 算法

传统的蒙特卡洛贪心算法是从种子节点出发进行大量的模拟计算后得出种子节点的影响范围。而 RIS 算法 [32] 采用反向蒙特卡洛算法，为了避免从种子节点出发进行的大量模拟计算，RIS 算法会从图上随机选取节点，然后从这些节点出发进行反向影响力传播的模拟，同时将碰到的节点加入到反向可达集合中。经过大量的反向模拟操作后，如果一个节点经常出现在反向可达集合中，那么该节点就是一个影响力大的节点。RIS 的理论时间复杂度已接近线性时间，同时仍有着 $1-1/e-\epsilon$ 的近似比保证。



(a) 社交网络 G　　(b) 随机图 g_1

图 2.1　RIS 算法示意图.

考虑独立级联模型，我们试图从图2.1中 (a) 所示的社交网络 G 选取一个影响力最大的节点。RIS 首先会产生足够多的反向可达集合，其产生过程对应到图2.1中可以分为两步：首先在社交网络 G 中随机选取一个节点，假设我们随机选取的节点为 v_1；然后对于独立级联模型，网络中的每条边 e 都对应有一个激活概率 $p(e)$，对于社交网络 G 中的每一条边 e，我们会以 $1-p(e)$ 的概率将它删掉，最终得到随机图 g_1。因为 g_1 中只有 v_1 和 v_4 可以到达 v_1，因此节点 v_1 对应的反向可达集合为 $R_1=\{v_1,v_4\}$。假设 RIS 通过上述过程得到了另外三个反

万方数据

向可达集合 $R_2=\{v_2\}$，$R_3=\{v_3\}$，$R_4=\{v_4\}$，因为 v_4 在反向可达集合中出现频率最高，所以 RIS 会认为 v_4 是影响力最大的节点。

2.2.3.2 TIM 和 IMM 算法

在 RIS 算法的基础上又有人提出了 TIM 和 IMM 算法。TIM [33] 和 IMM[18] 算法的基本思想和 RIS 一致，都是借助反向可达集合来寻找影响力最大的节点集合。RIS 虽然从理论上给出了接近线性的时间复杂度，但因为其时间复杂度中隐含着很大的常数项，所以难以扩展到大规模的网络中去。TIM 和 IMM 算法就是为了减小 RIS 中常数项带来的开销而提出的。RIS 常数项过大的原因是对需要的反向可达集合数目估计不准。RIS 为了保证结果的准确性需要产生足够多的反向可达集合，且产生的反向可达集合之间存在关联性。为了减弱关联性带来的影响，RIS 简单的将产生的反向可达集合数目设置的特别大，因此导致算法开销过大。TIM 和 IMM 从理论上分析了反向可达集合数目的下界，并巧妙地设计反向可达集合的产生方案，最终大大提高了算法的运行效率，实验证明，IMM 算法已超越了启发式算法的运行速度，同时还有着准确性上的理论保证。并且 IMM 也适用于独立级联、线性阈值和更广的触发模型。

2.3 随机游走

随机游走是一个数学统计模型，它描述的是一个由不同状态组成的随机序列。例如连续抛一枚质量均匀的硬币五次，第一次抛出的结果视为随机游走的出发点，那么正反面先后出现的次序就可为视为一个随机游走过程，该随机游走有两个状态，从一个状态到另一个状态的转移概率为 0.5，并且一个状态会以 0.5 的概率回到自身。可以看出随机游走的状态序列本质是一个马尔科夫链，马尔科夫链的一个性质就是无记忆性，即随机游走的下一步只与当前状态有关，与时间序列中它之前的状态无关。随机游走最主要的概念是转移概率，即从一个状态到另一个状态的概率。对应到图中，就是从当前节点选择图中另一节点作为下一步的概率。考虑图中最简单的随机游走，如果当前位于节点 u，那么它会随机均匀地选择一个邻居节点 v 作为下一步，因此对应的转移概率为：

$$p_{uv}=\begin{cases}\frac{1}{d_u}, v\in N(u),\\ 0, v\notin N(u).\end{cases} \tag{2.4}$$

其中，d_u 表示节点 u 的度数，$N(u)$ 表示节点 u 的邻居集合。

2.3.1 随机游走的应用

随机游走在图分析和图计算中有着广泛应用。本节我们从计算节点相似性，大图上采样和推荐系统三个方面来介绍随机游走的应用场景。由于考虑用户活

万方数据

动的社交网络可以抽象成超图，所以最后我们介绍超图中的随机游走，借它来帮助我们解决社交活动网络中的影响力最大化问题。

2.3.1.1　节点相似性

节点相似性有着广泛的应用，万维网中不同网页之间的超链接，语料系统中不同文章之间的引用，推荐系统中用户对产品的偏好，这些都可以作为衡量两个对象是否相似的基础信息。一个常见的应用是我们可以根据当前正在浏览的文档，利用相似性在一个语料库或者万维网中寻找其他相关文档。

SimRank[34] 是一种基于图的拓扑结构信息来衡量任意两个对象间相似程度的模型。其核心思想为：如果两个对象被相似或相同的对象引用，那么这两个对象也应该相似。例如对于网页来说，如果两个网页被很多相同的网页同时引用，那么我们就可以认为这两个网页是相似的。

SimRank 的定义为：

$$s(u,v)=\begin{cases}\frac{c}{|\delta(u)||\delta(v)|}\sum_{u'\in\delta(u),v'\in\delta(v)}s(u',v'),u\neq v,\\ 1,u=v.\end{cases}\tag{2.5}$$

由 SimRank 的定义可以看出，其通过递归利用了整个图的结构信息来表征两个节点之间的相似性，因此有着很好的准确性。但依赖于其他节点相似度的特性也给 SimRank 的计算带来很大的困难，尤其当整个图的结点数很多时，SimRank 计算具有很高的时间复杂度。

SimRank 可以利用基于随机游走的蒙特卡洛算法来计算，考虑图中两结点间的 SimRank 相似度 $s(u,v)$，基于随机游走，其有着这样的物理含义：对于两个分别从节点 u 和节点 v 同时出发的随机游走，在随机游走的过程中，他们会随机均匀地从当前所处节点的邻居节点中选择一个作为下一步。假设他们在图中某一个节点相遇时所走的步数为 $\tau(u,v)$，那么 $s(u,v)$ 和 $\tau(u,v)$ 有着以下对应关系：

$$s(u,v)=E(c^{\tau(u,v)}).\tag{2.6}$$

基于随机游走计算给定结点对 SimRank 相似度其时间复杂度为 $O(|V|)$；同时给定一个节点 u，基于 SimRank 查找与 u 最相关的 k 个节点，其时间复杂度可以降为 $O(kRT)$[35]，其中 R，T 均为随机游走过程中设置的常数信息。随机游走方法引入了随机性，因此准确性上不如传统的迭代法，但对于一些实际应用来说，我们往往只关注最后的排序信息，此时随机游走方法有着很强的实用价值。

2.3.1.2　图上的采样

社交网络中的一个重要问题是分析节点特征和网络结构特点。对于社交网络中的用户和研究者来讲，网络数据是属于网络运营商的，用户和研究者无权

万方数据

获得整个网络的信息，但其可以通过网络运营商提供的公共应用编程接口来爬取网络中的信息。不过出于保护用户隐私和商业机密的需求，这种接口的功能往往又是受到限制的，因此大规模爬取网络信息非常困难，且面对规模如此庞大的当今社交网络，我们只能通过从网络中采样少量节点的信息来估计整个网络的节点特征和结构特点。

很多情况下，用户和研究者们往往无法获得网络的实际规模，比如网络中的实际用户数；而且用户 ID 在空间上的分布往往很稀疏，比如微博用户的 ID 往往由 10 位数字构成（2014 年 3 月前，其 ID 分布为从“1000000000”到“5058913818”），但其中有效用户的比例可能不到 20%，并且当我们将采样数据做了某些限定后（比如只研究某个指定城市的用户），其空间分布会变得更加稀疏，这就导致了简单的均匀采样在很多场景中无法使用。

基于随机游走的采样是大规模网络采样中非常普遍的做法。随机游走采样从一个节点出发，将随机游走过程中碰到的节点作为采样节点。随机游走采样结果往往是有偏的，因此我们还需要对采样结果进行无偏化处理等操作来得到最终的采样结果。随机游走采样方法很多，常见的有以下几种：

- **简单随机游走**: 简单随机游走 [36] 采样从网络中的一个节点出发，在随机游走的每一步，它会随机均匀地选择当前节点的一个邻居作为下一跳的起始点。简单随机游走有着很多限制，比如要求网络连通，容易陷在网络的局部结构中，收敛概率和节点度数直接相关，因此会远远偏离均匀分布等。简单随机游走在采样结束后会有一个调整权重的过程，该调整过程会将有偏的采样结果纠正为无偏的。

- **可回到自身的随机游走**: 简单随机游走的采样结果是有偏的，这大大降低了随机游走采样的效率。可回到自身的随机游走 [37] 能使得采样节点自动达到无偏。此时对于随机游走的每一步来说，其不再是随机均匀地选择一个邻居节点作为下一步。假设当前随机游走位于节点 u，此时其转移概率为：

$$p_{uv} = \begin{cases} 1/d_u \times \min\{1, d_u/d_v\}, v \in N(u), \\ 1 - \sum_{u \neq w} p_{uw}, u = v, \\ 0, otherwise. \end{cases} \tag{2.7}$$

  可回到自身的随机游走虽然可以直接产生均匀的采样，但其随机游走过程中回到自身的可能性非常大。在实际网络中，其回到自身的概率经常会远远大于选择其邻居的概率，这就导致了随机游走总是困在自身而无法进行有效采样，采样过程会很慢。

- **可跳跃的随机游走**: 简单随机游走还有着容易陷入网络局部结构的缺点，这是因为网络中存在着社区结构，社区与社区之间的连接紧密程度远小于

万方数据

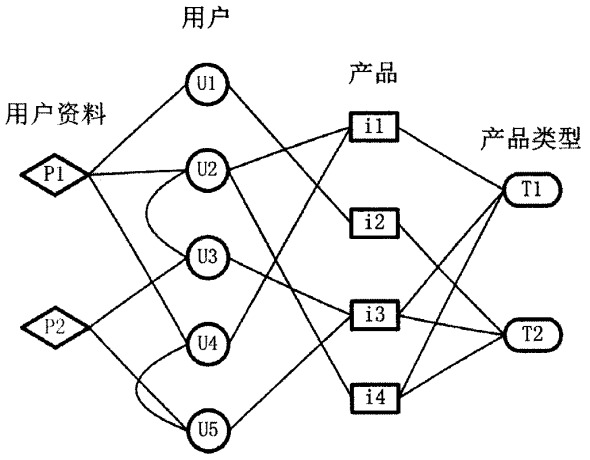


图 2.2　考虑社交信息的个性化推荐

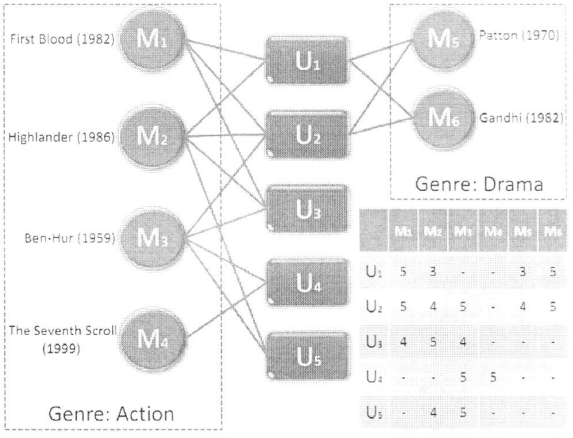
社区内连接的紧密程度。可跳跃的随机游走 [38] 在简单随机游走的过程中引入了均匀采样的过程，其每一步有着两个选择，一是以一定概率选择当前节点的邻居节点，二是以另一个概率从整个网络中随机均匀选择一个节点。因此带跳跃的随机游走可以避免其陷入局部结构的情况。带跳跃的随机游走的缺点是网络中节点 ID 的排布空间往往是很稀疏的，因此在从整个网络选取节点时可能需要做多次尝试后才能真正选到有效节点，从而降低了它的效率。

2.3.1.3　个性化推荐

个性化推荐是根据用户的兴趣特点和购买行为，向用户推荐可能感兴趣的信息和商品。大多数个性化推荐系统是基于用户对商品的偏好来进行推荐的，比如根据用户的浏览信息，用户对商品的评价等。社交网络中的用户存在着同质性，也就是说，好友之间在兴趣爱好上存在一定的相似性，因此好友的喜好从一定程度上反映了所研究用户的喜好。另外，推荐系统面临的一个重要问题是冷启动，这是因为新加入的用户购买记录和浏览信息过少，导致推荐信息不准确。如果我们能够充分利用用户的社交关系，那么对于提高推荐准确性，解决冷启动问题都会有着重要意义。

我们用一个简单例子 [39] 来说明随机游走在考虑了社交关系的个性化推荐系统中的应用。考虑图2.2所示的网络，其中节点包括用户，产品条目，产品类型和用户资料。节点间存在着以下四种关系：用户间的好友关系，用户间在资料上的相关性，用户对产品的评价，产品在类型上的相关性。各边上的权重设置可以分为以下两类：用户和产品条目之间的权重和用户对产品的评分直接相关，评分越高权重越大；对于其他边，如果关系存在，权重就设置为 1。我们可以利用随机游走，比如以一定概率回到自身的随机游走，从所研究用户出发，

万方数据



	M1	M2	M3	M4	M5	M6
U1	5	3	-	-	3	5
U2	5	4	5	-	4	5
U3	4	5	4	-	-	-
U4	-	-	5	5	-	-
U5	-	4	5	-	-	-

图 2.3　长尾推荐

随机游走达到稳态时各产品条目对应的概率分布就可以视为对用户进行推荐时的排序信息。

随机游走还可以用于长尾推荐 [40]。相对于广受欢迎的产品，冷门产品的市场竞争没那么激烈，因此利润率往往更高，并且冷门产品的成功推荐还可以带来客户对商家的信任感，从而带动客户的二次消费，间接促进热门产品的销售量。因此相对于推荐热门产品，深入发掘用户喜好，准确定位用户需求是很有实际价值的。考虑图2.3所示的电影评价网络，其中节点包含用户和电影条目，网络中的边代表是用户对电影的评分信息，边上的权重即为用户对电影的评分。利用随机游走，给定一个用户，我们可以求出从其他节点（电影条目）出发的随机游走碰到该用户点时所走的平均步长，并将该步长视为排序信息用于个性化推荐，平均步长越大，该用户喜欢该电影的可能性就越小。在该示例中，根据右下角所示的评分信息，对于用户 U5，利用随机游走进行长尾推荐会将电影 M4 推荐给 U5，但普通的推荐方法往往会推荐电影 M1，很显然 M4 并不是热门电影。

2.3.2　超图上的随机游走

超图是对简单图的扩展。简单图中的节点信息是成对存在的，因此每条边由两个节点构成，而超图中的边可以包含两个以上的节点，它可以反映两个以上节点之间的相互关系，超图中的边被称为超边。如图2.4所示是一个由 7 个节点构成的网络，其中存在 4 条超边，可以看出超边中的节点数从 1 到 3 不等，我们可以认为由两个节点组成的超边，比如 e_2，就是社交网络中普通的二元关系，像好友关系，关注等。而两个以上节点组成的超边，比如 e_1 和 e_3，则可以认为是社交网络中的用户共同参加了某个群组或者加入了某个话题等所形成的关系。

对于超图中的随机游走 [41]，因为其中的边可能包含两个以上的节点，因

万方数据

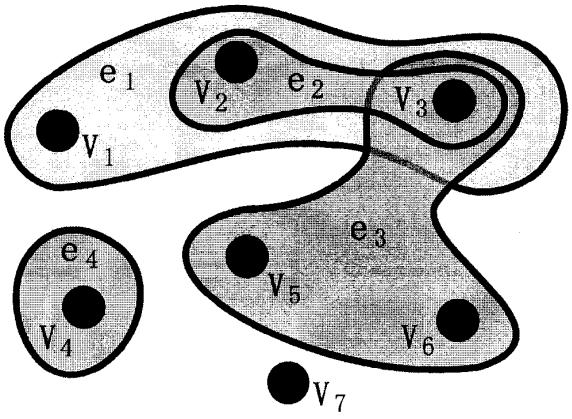


图 2.4　超图示例

此随机游走的每一步是分为两个阶段完成的：假设随机游走当前位于节点 u，首先我们会以一定概率选择一条包含节点 u 的超边；假设我们第一步选择了超边 e，接下来我们会再以一定的概率选择超边 e 中的一个节点做为我们的下一步。考虑超图 $G(V,\mathcal{E})$，其中 V 表示超图中的节点集合，$\mathcal{E}$ 表示超边的集合。我们规定用 $h(u,e)$ 作为反映节点 u 是否属于超边 e 的指示函数，当 $u \in e$ 时，$h(u,e)=1$；否则为 0。那么我们可以定义超图中节点 u 和超边 e 的度数分别为：

$$d(u)=\sum_{e\in\mathcal{E}} h(u,e). \tag{2.8}$$

$$\delta(e)=\sum_{u\in V} h(u,e). \tag{2.9}$$

考虑超图中简单的随机游走，每一步会先随机均匀地选择一条超边，然后再随机均匀地选择一个节点作为下一步。此时从 u 到 v 的转移概率为：

$$p_{uv}=\frac{1}{d(u)}\sum_{e\in\mathcal{E}(u)\cap\mathcal{E}(v)}\frac{1}{\delta(e)}. \tag{2.10}$$

其中 $\mathcal{E}(u)$ 表示包含节点 u 的超边集合。

2.4　本文的研究意义

本文的第一个研究意义是将影响力传播模型扩展至有用户活动存在的场景。上文已经提到目前的影响力最大化方法主要分为两类，一类是最大化基于网络拓扑的中心性方法，另一类则是基于影响力传播模型来设计影响力最大化算法。中心性方法缺乏对网络中用户实际行为数据的考虑，因此在刻画用户影响力时往往不够准确。影响力传播模型既考虑了网络拓扑，又反映了用户的实际行为，是目前研究的主流，但其只考虑了网络中用户的直接关系，对于存在大量用户

万方数据

活动的社交网络，我们需要对影响力传播模型进行扩展以适应有用户活动存在的场景。

本文的另一个研究意义是在扩展后的影响力传播模型下提出了快速的影响力最大化算法。单纯考虑直接用户关系的影响力最大化问题已被证明是 NP 难问题，将大量的用户活动考虑在内后，该问题将变得更加复杂；同时大量用户活动的存在使得底层的关系网络变得极为稠密，因此传统影响力最大化算法的时间开销会变得非常大。本文利用超图上的随机游走来近似影响力在社交活动网络中的传播过程，并定义了影响力中心性指标来近似节点集合的影响力。通过基于蒙特卡洛的近似算法，本文实现了对影响力中心性的快速计算，并进一步提出了贪心算法以及若干优化技术，实现了对影响力最大化问题的快速求解。

2.5　本章小结

本章首先介绍了一些影响力最大化的相关研究。节点影响力可以基于网络拓扑结构来刻画，常见的有度中心性、接近中心性、中介中心性和 PageRank 等。这些指标虽然可以从一定程度上反映节点在网络中影响力的大小，但往往不够准确。影响力传播模型既考虑网络拓扑，又可以反映网络中用户的实际行为，因此能更准确的刻画节点影响力。给定影响力传播模型就可以设计相应的影响力最大化算法，其主要分为贪心算法、启发式算法和反向蒙特卡洛算法。目前最先进的反向蒙特卡洛算法 IMM 运行速度上已经超过了启发式算法，而且有着一定的理论保证。本章还介绍了随机游走，这是我们解决社交活动网络中的影响力最大化问题所用到的理论工具，在介绍了随机游走的一些常见应用后，我们又将其扩展到超图上进行了简单介绍。最后本章对本文的研究意义进行总结，即将影响力传播模型扩展至有用户活动存在的场景，并在该场景下提出了快速的影响力最大化算法。

万方数据

第三章　社交活动网络中的影响力最大化

本章摘要： 本章具体介绍我们的研究工作。当今的社交网络不仅包含直接的用户关系，也有着各种用户活动所产生的间接关系。这些用户活动产生的间接关系对于影响力的传播至关重要。为了解决将用户活动考虑在内的影响力最大化问题，本文构建了超图模型对社交活动网络进行了建模，并将传统的影响力传播模型扩展至其中。基于随机游走理论，本文定义了影响力中心性指标来近似节点集合的影响力，并提出了基于蒙特卡洛的近似算法，实现了对影响力中心性的快速计算。最后利用子模函数的性质，本文进一步提出了贪心算法和若干优化技术，实现了对影响力最大化问题的快速求解。

3.1　问题描述

传统的社交网络只考虑直接的用户关系，因此用简单图刻画即可，在社交活动网络中，我们不仅要考虑直接的用户关系，也要考虑用户活动产生的间接关系，因此我们使用超图来刻画这种有多种用户关系存在的网络。同时，传统的影响力最大化问题也是针对简单图的，因此必须根据超图的特点进行扩展。

3.1.1　社交活动网络模型

我们使用超图 $G(V, E, \mathcal{E}_1, ..., \mathcal{E}_l)$ 来刻画社交活动网络。在超图模型中，V 表示网络中所有用户的集合；E 表示直接用户关系所形成的边的集合，比如微博中的关注关系；$\mathcal{E}_i$ 表示第 i 类超边的集合，每一个超边对应着一个活动的参与者所形成的集合，比如微博中的用户共同参加了关于某部电影的话题。我们用元组来表示每一条超边。对于图 1.3所示的社交活动网络来说，假设活动 a 属于第一类活动，那么 $\mathcal{E}_1 = \{(1,2,3,5)\}$；同理，活动 b 属于第二类，所以 $\mathcal{E}_2 = \{(1,6)\}$。此时图1.3可以表示为 $G(V, E, \mathcal{E}_1, \mathcal{E}_2)$，其中 $V = \{1,2,3,4,5,6\}$，$E = \{(1,2),(2,1),(2,3),(3,2),(3,4),(4,3),(4,5),(4,6),(5,4),(6,4)\}$。为了方便，我们用 $N(j)$ 表示用户 j 的直接邻居集合，即 $N(j) = \{i|(i,j) \in E\}$，$M_e(j)$ 表示除了 j 以外属于超边 e 的其他用户集合，即 $M_e(j) = \{i|i \in e, \&i \neq j\}$；另外我们也定义 $\mathcal{E}_t(j)$ 表示包含 j 的第 t 类超边集合，即 $\mathcal{E}_t(j) = \{e|e \in \mathcal{E}_t \& j \in e\}$。对于图 1.3，$N(1) = \{2\}$; $M_e(1) = \{2,3,5\}$，其中 $e = (1,2,3,5)$；$\mathcal{E}_1(1) = \{(1,2,3,5)\}$。

万方数据

3.1.2　社交活动网络中的影响力最大化问题定义

回顾之前介绍的影响力传播模型，对于独立级联模型来说，用户可以处于两种状态中的一个：接受和未接受。首先我们选取一个用户集合作为种子集合，并将它们全部激活。对于一个刚被激活的用户 i，它会以一定的概率 q_{ij} 来激活其邻居用户 j（$j \in N(i)$）使其处于接受状态。设置激活概率是一个复杂的学习过程，为了方便，很多工作会采用一种均匀设置的方法 [9, 18, 27, 33, 42]，比如 $q_{ij} = \frac{1}{d_j}$。当邻居 j 被激活以后，它会继续尝试激活未被激活的邻居节点，该过程一直持续至没有新的节点被激活为止。

独立级联模型因其简单有效且确实符合一些实际应用的特点（比如新消息在社交网络中的传播或病毒在人际间的传播），因此是研究的最为深入，广泛的模型。此处我们也选择独立级联模型进行扩展以使其适应有用户活动存在的情况。其关键问题是如何在考虑用户活动的情况下定义用户间的影响，这可以基于以下三个准则。

- 实际生活中，用户做出的（购买）决策是受到多方面因素影响的，但可以简单归纳为自身因素和外界影响。我们研究的是网络中的用户通过直接关系或者间接关系被周围人影响而做出决策的情况。我们定义用户被周围人影响的概率为 $c(0 < c < 1)$，并称之为衰减系数。假设网络中存在 l 类活动，我们定义 α_{jt}（其满足 $0 \leqslant \alpha_{jt} \leqslant 1$ 并且 $0 \leqslant \sum_{t=1}^{l} \leqslant 1$）为用户 j 受到的来自第 t 类活动影响的比例，并称之为活动权重。显然 $1 - \sum_{t=1}^{l} \alpha_{jt}$ 就代表用户 j 受到的来自其直接邻居影响的比例。

- 对于用户 j 受到的来自于其直接邻居的影响，我们定义来自于每个邻居 i（$i \in N(j)$）的影响占比为 u_{ij}，且 u_{ij} 应当满足 $0 \leqslant u_{ij} \leqslant 1$ 和 $\sum_{i \in N(j)} u_{ij} = 1$。

- 对于用户 j 受到的来自于第 t 类活动的影响，我们定义来自于每个活动 a 的影响占比为 v_{aj}，同理，v_{aj} 应满足 $0 \leqslant v_{aj} \leqslant 1$ 和 $\sum_{a \in N_t(j)} v_{aj} = 1$。考虑到同一个活动中往往存在多个用户，我们又定义来自于同一活动 a 的每个用户 i 的影响占比为 u_{ij}^{a} $(i \in N(a) \backslash \{j\})$，且满足 $\sum_{i \in N(a) \backslash \{j\}} u_{ij}^{a} = 1$。

为了简单起见，我们采取均匀的设置方法。均匀设置方法是独立级联模型中最常采用的方法。具体到我们的模型中分为三种情况：第一，对于用户 j 来说，来自同一类活动的影响是均匀的，即 $v_{aj} = 1/|\mathcal{E}_t(j)|$；第二，对于用户 j 来说，来自其直接邻居的影响是均匀的，即 $u_{ij} = 1/|N(j)|$；最后，对于用户 j 来说，来自同一活动中的其他用户的影响是均匀的，即 $u_{ij}^{a} = 1/|M_e(j)|$。同时我们也指出，我们的计算框架可以经过简单的扩展来适应非均匀设置的情况。现在我们可以定义超图（社交活动网络）中用户 i 对用户 j 的影响为：

$$g_{ij} = c \times \left(\frac{1 - \sum_{t=1}^{l} \alpha_{jt}}{|N(j)|} \times \mathbf{1}_{\{i \in N(j)\}} + \sum_{t \in [1,l]} \sum_{e \in \mathcal{E}_t(j)} \frac{\alpha_{jt}}{|\mathcal{E}_t(j)|} \times \frac{1}{|M_e(j)|} \times \mathbf{1}_{\{i \in M_e(j)\}} \right). \tag{3.1}$$

万方数据

公式 (3.1) 的第一部分表示来自于直接关系的影响，其第二部分表示由于用户活动而受到的影响，即来自于间接关系的影响。

现在我们可以基于独立级联模型来考虑社交活动网络中的影响力最大化问题。我们将其表示为 **IMP(SAN)**：

定义 3.1.1. IMP(SAN): 给定社交活动网络 $G(V, E, \mathcal{E}_1, ..., \mathcal{E}_l)$，影响力传播模型和参数 α_{jt}、k，找出由 k 个节点组成的集合 $S(k)$，使得这 k 个点在该影响传播模型下的影响力延展度 $\sigma(S(k))$ 最大.

从该问题的描述我们可以看出来将活动考虑在内后，影响力最大化问题的主要不同是多了活动的权重这些参数。

3.2　考虑用户活动的影响力最大化方法

社交活动网络中的影响力最大化是一个 NP 难问题 [25]，即使在给定影响力传播模型下计算节点集合的影响力延展度也是 #P 难问题 [26]，将大量的用户活动考虑在内后，问题会变得更加复杂。为了解决这些困难，我们首先定义了一个中心性来近似用户集合的影响力延展度，该中心性是基于超图中的随机游走定义的，我们称之为影响力中心性，此时我们面临的就是一个中心性最大化问题。通过精心设计高效的中心性计算方法和贪心优化技巧，我们就能得到影响力最大化问题的近似最优解。

3.2.1　考虑用户活动的随机游走

对于一个简单的无权图 $G(V, E)$ 来说，简单的随机游走过程如下：当随机游走位于节点 i 时，它会以均匀的概率的选择一个节点 i 的邻居节点 j 作为下一步随机游走的出发点。如果我们用 $Y(t)$ 来表示随机游走在第 t 步时对应的位置，那么 $\{Y(t)\}$ 就构成了一个马尔科夫链。对于一个节点 j，如果 $(i, j) \in E$, 那么转移概率 $p_{ij} = 1/N(i)$；否则转移概率为 0。

超图中的边可能包含两个以上的节点，因此不能在随机游走的每一步直接选择一个邻居节点作为其下一步。因此超图中随机游走的每一步实际上是分为两个阶段完成的：假设随机游走当前位于节点 u，首先我们会以一定概率选择一条包含节点 u 的超边；假设我们第一步选择了超边 e，接下来我们会再以一定的概率选择超边 e 中的一个节点做为我们的下一步。

社交活动网络可以建模成一个超图 $G(V, E, \mathcal{E}_1, ..., \mathcal{E}_l)$，其超边可以分为不同的类型，E 表示直接用户关系所形成的边的集合，$\mathcal{E}_i$ 表示第 i 类超边的集合。考虑用户活动的随机游走对应着超图 $G(V, E, \mathcal{E}_1, ..., \mathcal{E}_l)$ 上的随机游走，因此也分为两个阶段：

- **第一阶段:** 对于位于节点 i 的随机游走，它会以一定的概率选择一条与其邻接的超边。具体来讲，根据超图中影响力传播模型的描述，用户 i 受到的来自于

万方数据

第 t 类活动影响的权重为 α_{it}，因此我们将选择第 t 类超边的概率设置为 α_{it}。同时根据 §3.1.2中的均匀设置规则一，一个用户受到的来自同一类活动的影响是均匀的，因此我们会从第 t 类超边中随机选择一条超边作为第二阶段的开始。因此对于当前位于 i 的随机游走，它会以概率 $\frac{\alpha_{it}}{|\mathcal{E}_t(i)|}$ 来选择第 t 类超边中的某一个 e。

• **第二阶段：** 当第一阶段已经选择了一条超边 e，接下来的随机游走会从该超边中随机选择一个节点作为随机游走的下一步。我们此处使用无回退的随机游走，且根据 §3.1.2中的均匀设置规则三，对于一个位于节点 i 的随机游走来说，我们会从超边 e 中随机均匀地选择一个节点 j（$j \neq i$）作为下一步，对应的概率为 $1/|M_e(i)|$。

根据以上随机游走的规则，我们可以得到从 i 到 j 的转移概率如下所示，且转移概率和用户间影响的关系为 $g_{ji} = c \times p_{ij}$。

$$p_{ij} = \frac{1-\sum_{t=1}^{l}\alpha_{it}}{|N(i)|} \times \mathbf{1}_{\{j\in N(i)\}} + \sum_{t\in[1,l]}\sum_{e\in\mathcal{E}_t(i)} \frac{\alpha_{it}}{|\mathcal{E}_t(i)|} \times \frac{1}{|M_e(i)|} \times \mathbf{1}_{\{j\in M_e(i)\}}. \tag{3.2}$$

考虑图 1.3所示的社交活动网络，我们认为活动 a 为第一类活动，活动 b 为第二类活动。用户 1 两个活动都有参加，我们设置 $\alpha_{11} = 0.5$，$\alpha_{12} = 0.3$；对于用户 2，3，5，其只参加了活动 a，我们设置 $\alpha_{j1} = 0.5$，$\alpha_{j2} = 0$，$j \in \{2,3,5\}$；用户 6 只参加了活动 b，我们设置 $\alpha_{61} = 0$，$\alpha_{62} = 0.5$；用户 4 未参加任何活动，因此 $\alpha_{41} = 0$，$\alpha_{42} = 0$。由此根据式 (3.2) 我们可以得到超图中随机游走的转移概率矩阵如下：

$$\mathbf{P} = \begin{pmatrix} 0 & 11/30 & 1/6 & 0 & 1/6 & 3/10 \\ 5/12 & 0 & 5/12 & 0 & 1/6 & 0 \\ 1/6 & 5/12 & 0 & 1/4 & 1/6 & 0 \\ 0 & 0 & 1/3 & 0 & 1/3 & 1/3 \\ 1/6 & 1/6 & 1/6 & 1/2 & 0 & 0 \\ 1/2 & 0 & 0 & 1/2 & 0 & 0 \end{pmatrix}. \tag{3.3}$$

3.2.2　影响力中心性

为了解决社交活动网络中的影响力最大化问题，我们首先面临的就是如何有效计算给定节点集合的影响力延展度，这本身是一个 #P 难问题，因此我们只能采用近似算法来提高计算效率。为此，我们首先定义了一个称之为影响力中心性的指标 I(S)，它基于超图上的随机游走过程而来，可以近似集合 S 在独立级联模型下的影响力延展度。其定义如下：

$$I(S) = \sum\nolimits_{j\in V} h(j,S). \tag{3.4}$$

万方数据

其中 $h(j,S)$ 表示从节点 j 出发按一定规则进行随机游走碰到 S 中点的概率，它可以指示影响力由 S 传播到 j 的难易程度。因此我们可以用 $h(j,S)$ 近似表示集合 S 对单个节点 j 的影响力，并称之为衰减命中概率。对从所有节点出发进行随机游走的衰减命中概率求和就可以近似集合 S 的影响力延展度。$h(j,S)$ 的定义如下：

$$h(j,S)=\begin{cases}\sum_{i\in V} cp_{ji}h(i,S), j\notin S,\\ 1, j\in S.\end{cases} \tag{3.5}$$

其中，c 对应 §3.1.2中的衰减系数，p_{ji} 是从 j 到 i 的转移概率。

为了解决社交活动网络中的影响力最大化问题，我们用影响力中心性 $I(S)$ 来近似集合 S 在考虑活动时的影响力大小。此时我们的目标就可以转化为寻找使得 $I(S)$ 最大的用户集合。也就是说，我们可以借助中心性最大化问题来解决社交活动网络中的影响力最大化问题。该中心性最大化问题的定义如下：

定义 3.2.1. CMP: 给定超图 $G(V,E,\mathcal{E}_1,...,\mathcal{E}_l)$ 和每个用户 $j\in V$ 对应的活动权重 α_{jt}，其中 $t\in[1,l]$，找出由 k 个节点组成的集合 S，使得集合 S 所对应的影响力中心性 $I(S)$ 最大.

接下来我们首先介绍如何利用超图上的随机游走来近似计算 $h(j,S)$，在解决 $h(j,S)$ 的计算问题后，我们就可以用简单的贪心算法来解决影响力中心性最大化问题。在简单贪心算法的基础上我们还会探讨如何进一步优化贪心算法的效率。

3.2.3　中心性计算

我们注意到解决影响力中心性最大化问题的一个关键点在于如何快速有效计算该中心性，即如何快速计算 $h(j,S)$。在大规模社交活动网络中准确计算 $h(j,S)$ 是一件非常困难的事情：基于式 (3.5) 中的递归定义我们可以看到计算 $h(j,S)$ 需要我们对所有 $h(i,S)$，其中 $i\in V$，进行计算，从而使得计算 $h(j,S)$ 的时间开销非常巨大。因此我们采用了近似计算的做法。

近似计算的大致框架如下：首先我们将 $h(j,S)$ 的递归表达转换成矩阵表达，该矩阵表达涉及到矩阵求逆；然后我们将该矩阵表达展开成无穷项相加收敛的形式，并取其前 L 项来近似 $h(j,S)$，由于我们只需对这前 L 项来进行计算，因此可以减少计算开销；最后我们利用超图上的随机游走来解释每一项的含义，并借助蒙特卡洛算法来完成近似计算。

我们首先将式 (3.5) 中的递归表达转换为矩阵表达。

定理 3.2.1. *式 (3.5) 中的衰减命中率 $h(j,S)$ 可以表达如下：*

$$h(j,S)=ce_j^{\mathsf{T}}(\mathbf{I}-c\mathbf{Q})^{-1}\mathbf{Q}'\mathbf{e},\quad (\text{其中} j\notin S). \tag{3.6}$$

$\mathbf{Q}$ 是一个 $(|V|-|S|)\times(|V|-|S|)$ 的矩阵，它描述随机游走时 $V-S$ 中任意两个节点之间的转移概率。$\mathbf{Q}'$ 是一个 $(|V|-|S|)\times|S|$ 的矩阵，它描述随机游走时从

万方数据

$V-S$ 中的一个节点到 S 中的一个节点的转移概率。$\mathbf{I}$ 是一个单位矩阵。$\boldsymbol{e}$ 是一个所有元素全为 *1* 的列向量。$\boldsymbol{e}_j$ 也是一个列向量，其元素中对应节点 j 的那个元素为 *1*，其余元素全为 *0*。

证明: 根据 $h(j,S)$ 在公式 (3.5) 中的定义, 我们可以得到

$$h(j,S)=\sum_{h=1}^{\infty}c^hP(j,S,h),\ \text{for}\ j\notin S.$$

其中 $P(j,S,h)$ 表示从 j 出发进行随机游走时在第 h 步碰到 S 中的点的概率。我们用 p_{iS} 来表示当前位于 i 的随机游走在下一步碰到 S 中点的概率。那么 $h(j,S)$ 就存在如下转换：

$$\begin{aligned}h(j,S) &= \sum_{h=1}^{\infty}c^hP(j,S,h)=\sum_{h=1}^{\infty}c^h\sum_{i\notin S}(\mathbf{Q}^{h-1})_{ji}p_{iS},\\ &= \sum_{i\notin S}\sum_{h=1}^{\infty}c^h(\mathbf{Q}^{h-1})_{ij}p_{iS}=\sum_{i\notin S}c(\mathbf{I}-c\mathbf{Q})^{-1}_{ji}p_{iS},\\ &= c\boldsymbol{e}_j^{\mathsf{T}}(\mathbf{I}-c\mathbf{Q})^{-1}\mathbf{Q}'\boldsymbol{e}.\end{aligned}$$

为了进一步解释公式 (3.6) 中 $\mathbf{Q}$ 和 $\mathbf{Q}'$ 的含义，我们在图3.1中给出了其在转移矩阵 $\mathbf{P}$ (3.3) 中的对应情况。

$$\mathbf{P}=\left(\begin{array}{cccc|cc}
0 & 11/30 & 1/6 & 0 & 1/6 & 3/10\\
5/12 & 0 & 5/12 & 0 & 1/6 & 0\\
1/6 & 5/12 & 0 & 1/4 & 1/6 & 0\\
0 & 0 & 1/3 & 0 & 1/3 & 1/3\\
\hline
1/6 & 1/6 & 1/6 & 1/2 & 0 & 0\\
1/2 & 0 & 0 & 1/2 & 0 & 0
\end{array}\right)$$

其中 $\mathbf{Q}$ 对应左上 4×4 块，$\mathbf{Q}'$ 对应右上 4×2 块。

图 3.1　当 $S=\{5,6\}$ 时，参数 $\mathbf{Q}$ 和 $\mathbf{Q}'$ 在转移矩阵中的对应情况

由于 $c\mathbf{Q}$ 对应的最大特征值小于 1，因此我们可以将式 (3.6) 进行展开，其展开形式如下：

$$h(j,S)=c\boldsymbol{e}_j^{\mathsf{T}}\mathbf{Q}'\boldsymbol{e}+c^2\boldsymbol{e}_j^{\mathsf{T}}\mathbf{Q}\mathbf{Q}'\boldsymbol{e}+c^3\boldsymbol{e}_j^{\mathsf{T}}\mathbf{Q}^2\mathbf{Q}'\boldsymbol{e}+\cdots. \tag{3.7}$$

因为衰减系数 c 满足 $0<c<1$，且对于 $n\geqslant0$ 有 $\boldsymbol{e}_j^{\mathsf{T}}\mathbf{Q}^n\mathbf{Q}'\boldsymbol{e}\leqslant1$，因此该展开形式是收敛的。为了减少计算上的开销，我们只取该展开形式的前 L 项来近似其结果，并表示如下：

$$h^L(j,S)=c\boldsymbol{e}_j^{\mathsf{T}}\mathbf{Q}'\boldsymbol{e}+c^2\boldsymbol{e}_j^{\mathsf{T}}\mathbf{Q}\mathbf{Q}'\boldsymbol{e}+\ldots+c^L\boldsymbol{e}_j^{\mathsf{T}}\mathbf{Q}^{L-1}\mathbf{Q}'\boldsymbol{e}. \tag{3.8}$$

我们可以对该截断形式带来的误差进行简单分析，其误差满足

$$0\leqslant h(j,S)-h^L(j,S)\leqslant\frac{c^{L+1}}{1-c}. \tag{3.9}$$

万方数据

根据截断形式和展开式之间误差表达，我们可以看到 $h^L(j,S)$ 以 c^{L+1} 的速度逼近 $h(j,S)$。这也就是说如果我们想要获取不大于 ϵ（$0 \leqslant \epsilon \leqslant 1$）的误差率，我们只需要取足够大的 L 即可，其满足 $L \geqslant \lceil \frac{\log \epsilon(1-c)}{\log c} \rceil - 1$。

现在我们可以使用蒙特卡洛算法来对 $h^L(j,S)$ 进行近似计算，该算法可以通过少量的随机游走来实现对 $h^L(j,S)$ 的准确近似。

式 (3.8) 中的项 $\boldsymbol{e}_j^T \mathbf{Q}^{t-1} \mathbf{Q}' \boldsymbol{e}$ 对应的随机游走含义如下：考虑超图中从点 j（$j \notin S$）出发的最长 L 步的随机游走。对于随机游走的每一步，如果当前位于节点 k（$k \notin S$），那么它会以 p_{ki} 的概率选择一个 k 的邻居 i，p_{ki} 的定义在式 (3.2) 给出。当随机游走碰到的点位于 S 中时就停止随机游走。我们用 $j^{(t)}$（$t = 1, ..., L$）来表示随机游走在第 t 步所处的位置，并定义如下所示的量：

$$X(t) = \begin{cases} 1, & j^{(t)} \in S, \\ 0, & j^{(t)} \notin S. \end{cases}$$

我们可以看出 $\boldsymbol{e}_j^T \mathbf{Q}^{t-1} \mathbf{Q}' \boldsymbol{e}$ 是从 j 出发的随机游走在第 t 步碰到 S 中点的概率。即：

$$\boldsymbol{e}_j^T \mathbf{Q}^{t-1} \mathbf{Q}' \boldsymbol{e} = E[X(t)]. \tag{3.10}$$

我们用式 (3.10) 来替换式 (3.8) 中的每一项，最终可以得到

$$h^L(j,S) = cE[X(1)] + c^2 E[X(2)] + \cdots + c^L E[X(L)]. \tag{3.11}$$

根据式 (3.11)，我们可以利用基于超图上随机游走的蒙特卡洛算法来近似计算 $h^L(j,S)$。特别的，对于每个满足 $j \notin S$ 的节点 j，我们可以设置 R 个从 j 出发的相互独立的随机游走。对于每一个从 j 出发的随机游走，如果当前位于节点 k，且节点 k 满足 $k \notin S$，那么就以 p_{ki} 的概率选择一个 k 的邻居节点 i。当随机游走碰到了 S 中的点或者达到了 L 步，那么该随机游走停止。我们定义这 R 个随机游走在第 t 步的位置分别为 $j_1^{(t)}, j_2^{(t)}, \ldots, j_R^{(t)}$，并用 $X_r(t)$ 来指示 $j_r^{(t)}$ 是否属于 S。更准确的说，当 $j_r^{(t)} \in S$ 时，$X_r(t) = 1$；否则，$X_r(t) = 0$。现在式 (3.11) 中的每一项 $c^t E[X(t)]$ 就可以被估计为

$$c^t E[X(t)] \approx \frac{c^t}{R} \sum_{r=1}^{R} X_r(t).$$

用其替换式 (3.11) 中的每一项，我们就可以有效近似 $h^L(j,S)$，我们将近似值表示为

$$\hat{h}^L(j,S) = \frac{c}{R} \sum_{r=1}^{R} X_r(1) + \frac{c^2}{R} \sum_{r=1}^{R} X_r(2) + \cdots + \frac{c^L}{R} \sum_{r=1}^{R} X_r(L). \tag{3.12}$$

我们可以证明 $\hat{h}^L(j,S)$ 是 $h^L(j,S)$ 的无偏估计.

定理 3.2.2. $\hat{h}^L(j,S)$ 是 $h^L(j,S)$ 的无偏估计.

万方数据

证明: 对 $E(\frac{1}{R}\sum_{r=1}^{R} X_r^{(t)})$ 进行简单变换可得

$$E(\frac{1}{R}\sum_{r=1}^{R} X_r^{(t)}) = \frac{1}{R}\sum_{r=1}^{R}(E(X_r^{(t)})) = E(X_r^{(t)}).$$

此时 $E(\hat{h}^L(j,S))$ 可以表达为

$$\begin{aligned} E(\hat{h}^L(j,S)) &= E(\frac{c}{R}\sum_{r=1}^{R} X_r^{(1)}) + E(\frac{c^2}{R}\sum_{r=1}^{R} X_r^{(2)}) + \ldots + E(\frac{c^L}{R}\sum_{r=1}^{R} X_r^{(L)}), \\ &= cE(X_r^{(1)}) + c^2E(X_r^{(2)}) + \ldots + c^LE(X_r^{(L)}), \\ &= h^L(j,S) \end{aligned}$$

即证 $\hat{h}^L(j,S)$ 是 $h^L(j,S)$ 的无偏估计.

```
input : 超图 G(V, E, ε_1, ..., ε_l)，节点 j 和节点集合 S
output: h^L(j, S) 的近似值
σ ← 0;
for r = 1 to R do
    i ← j;
    for t = 1 to L do
        产生随机数 x ∈ [0, 1];
        for T = 0 to l do
            if x ⩽ α_iT then
                E ← ε_T(i);
                break;
            end
            x ← x − α_iT;
        end
        从超边集合 E 中随机选择超边 e;
        i ← 从 {k|k∈e, k≠i} 中随机选择一个用户;
        if i ∈ S then
            σ ← σ + c^t/R;
            break;
        end
    end
end
return σ;
```

算法 3.1: 蒙特卡洛算法近似计算 $h^L(j,S)$

算法 3.1展示了用了蒙特卡洛算法来对 $h^L(j,S)$ 进行近似计算的过程。因为活动种类 l 往往很小，所以我们可以认为该近似算法的时间复杂度为 O(RL)。也就是说，我们可以用 O(RL) 的时间来近似计算 $h^L(j,S)$。由于计算 I(S) 时需要从所有节点 $j \in (V-S)$ 出发进行随机游走, 因此最终计算 I(S) 的时间开销就是 O(nRL)。用蒙特卡洛算法来计算 $h^L(j,S)$ 的最大优点是时间开销独立于图规模的大小，因此其可以扩展到很大规模的图计算中去。

万方数据

利用算法 3.1得到的 $\hat{h}^L(j,S)$ 是对 $h^L(j,S)$ 的近似值，近似算法的准确度取决于 R 的大小。为了确保我们的近似有效，需要我们估计出 R 的合理设置。这里我们利用 Hoeffding 不等式 [43] 推导出了 R 应该满足的下界值。

定理 3.2.3. *算法 (3.1) 的输出表示为 $\hat{h}^L(j,S)$, 我们可以得到*

$$P\{|\hat{h}^L(j,S) - h^L(j,S)| > \epsilon\} \leqslant 2L\exp(-2(1-c)^2\epsilon^2 R). \tag{3.13}$$

证明: 令 $X_1,...,X_R$ 表示 R 个独立的随机变量，且对于 $r = 1,\cdots,R$，其满足 $X_r \in \{0,1\}$。设置 $T = (X_1 + ... + X_R)/R$. 根据 Hoeffding 不等式，我们有 $P\{|T - E(T)| > \epsilon\} \leqslant 2\exp(-2\epsilon^2 R)$. 进一步利用此不等式我们可以得到

$$\begin{aligned}
& P\{|\hat{h}^L(j,S) - h^L(j,S)| > \epsilon\} \\
= \ & P\Big\{|\sum_{t=1}^{L}\frac{c^t}{R}\sum_{r=1}^{R}X_r(t) - \sum_{t=1}^{L}c^tE[X(t)]| > \epsilon\Big\}, \\
\leqslant \ & P\Big\{\sum_{t=1}^{L}|\frac{c^t}{R}\sum_{r=1}^{R}X_r(t) - c^tE[X(t)]| > \epsilon\Big\}, \\
\leqslant \ & \sum_{t=1}^{L}P\Big\{|\frac{c^t}{R}\sum_{r=1}^{R}X_r(t) - c^tE[X(t)]| > (1-c)c^t\epsilon\Big\}, \\
\leqslant \ & 2L\exp(-2(1-c)^2\epsilon^2 R).
\end{aligned}$$

根据定理 3.2.3，我们可以得出算法 3.1在满足 $R \geqslant \log(2L/\delta)/(2(1-c)^2\epsilon^2)$ 的情况下，会以大于 $1-\delta$ （$0<\delta,\epsilon<1$）的概率保证对 $h^L(j,S)$ 的近似误差小于 ϵ。

3.2.4　中心性最大化

在解决计算影响力中心性的问题后，我们就可以来设计算法去解决中心性最大化问题了。影响力中心性最大化问题的定义我们已经在 §3.2.2中给出。应该认识到，即使我们可以利用随机游走来快速近似计算 I(S)，要想直接在社交活动网络中找出使得 I(S) 最大的 k 个节点集合在计算上也是十分困难的。这是因为我们需要遍历所有 k 个节点的组合，这在当今社交网络规模如此庞大的情况下是不现实的。事实上，我们可以证明影响力中心性最大化问题是 NP 难问题。

定理 3.2.4. *影响力中心性最大化问题是 NP 难问题.*

证明: 首先我们介绍图论中的顶点覆盖问题，它是一个 NP 完全问题。顶点覆盖问题的描述如下：给定图 G(V, E) 和一个大于 0 的整数 k，我们需要确定在图中是否可以找出一个顶点集合 S 使得 $|S| \leqslant k$ 且对于每条边 $(i,j) \in E$，有 $\{i,j\} \cap S \neq \emptyset$。

万方数据

现在我们讨论影响力中心性最大化问题和顶点覆盖问题的映射关系。考虑这样一个判定性问题，在和顶点覆盖问题同样输入的情况下，即输入 $G(V,E)$ 和大于 0 的整数 k，我们需要判断是否存在一个顶点集合 S 满足 $|S| \leqslant k$ 且 $I(S) \geqslant (n-k)\times c+k$。我们可以说明当且仅当 $I(S) \geqslant (n-k)\times c+k$ 时，S 是一个顶点覆盖。为了得到这个结论，我们首先假设 S 是一个顶点覆盖，那么当 $i \in S$ 时，$h(i,S)=1$；当 $i \notin S$ 时，$h(i,S)=c$。对于每一条边 $(i,j) \in E$ 有 $\{i,j\} \cap S \neq \emptyset$，这也就是说 $I(S)=\sum_{i\in V} h(i,S)=k+\sum_{i\in(V-S)} h(i,S)=(n-k)\times c+k$。然后我们考虑 S 不是一个顶点覆盖的情况，那么图中至少存在一条边 (u,v) 满足 $\{u,v\} \cap S=\emptyset$。也就是说从 u 出发且经过 v 的随机游走在碰到 S 中的节点前至少要走 2 步，因此有 $h(u,S)=\sum_{j\in N(u)/v} cp_{uj}h(j,S)+cp_{uv}h(v,S)$。因为 $v \notin S$，所以 $h(v,S)<1$。所以 $h(u,S)<\sum_{j\in N(u)/v} cp_{uj}+cp_{uv}=c$，即 $I(S)=\sum_{i\in V} h(i,S)=k+\sum_{i\in(V-S)} h(i,S)<(n-k)\times c+k$。因此可以得出当且仅当 $I(S) \geqslant (n-k)\times c+k$ 时，S 是一个顶点覆盖的结论。这也就是说如果我们可以判定 $I(S) \geqslant (n-k)\times c+k$ 是否成立，那么我们就可以解决顶点覆盖问题；由于顶点覆盖问题是 NP 完全问题，因此该判定问题也是 NP 完全问题。另外，如果我们可以解决影响力中心性最大化问题，那么我们就能判定 $I(S) \geqslant (n-k)\times c+k$ 是否成立，现在该判定问题是 NP 完全问题，所以影响力中心性最大化问题难度上至少等价于 NP 完全问题，也就是说影响力中心性最大化问题是 NP 难问题。

为了有效解决影响力中心性最大化问题，我们只能设法找出其近似最优解。通过分析影响力中心性可以发现其满足子模函数 [16] 的性质。接下来我们会解释子模函数应该满足的条件以及该性质对于求近似最优解的好处，然后我们会介绍利用子模函数来求近似最优解的基本贪心算法，以及我们设计的对基本贪心算法的优化技术。现在我们就证明 $I(S)$ 是一个非递减的子模函数。

定理 3.2.5. *影响力中心性 $I(S)$ 是一个非递减的子模函数。*

证明: 我们首先展示 $I(S)$ 的非递减性。因为当 $j \in S$ 时，$h(j,S)=1$，所以 $I(S)$ 可以有如下表达：

$$I(S)=|S|+\sum_{j\in(V-S)} h(j,S).$$

将节点 $u \notin S$ 加入 S 后，影响力中心性的增量 $\Delta(u)=I(S\cup\{u\})-I(S)$ 可以有如下推导过程：

$$\begin{aligned}\Delta(u) &= \sum_{j\in V} h(j,S\cup\{u\})-\sum_{j\in V} h(j,S),\\ &= 1+\sum_{j\in(V-S\cup\{u\})} h(j,S\cup\{u\})-\sum_{j\in(V-S)} h(j,S),\\ &= 1-h(u,S)+\sum_{j\in(V-S\cup\{u\})}\Big[h(j,S\cup\{u\})-h(j,S)\Big].\end{aligned}$$

万方数据

根据式 (3.7) 中对 $h(j,S)$ 的定义及其对应的随机游走的含义，我们可以得出

$$h(j,S)=\sum_{h=1}^{\infty}c^{h}P(j,S,h),\ \text{for}\ j\notin S,$$

其中 $P(j,S,h)$ 表示从 j 出发的随机游走在第 h 步碰到 S 中点的概率。因此 $h(j,S\cup\{u\})-h(j,S)$ 可以表达为

$$\begin{aligned}
&h(j,S\cup\{u\})-h(j,S)\\
=&\sum_{h=1}^{\infty}c^{h}P(j,S\cup\{u\},h)-\sum_{h=1}^{\infty}c^{h}P(j,S,h)\\
=&\sum_{h=1}^{\infty}c^{h}P^{S\cup\{u\}}(j,\{u\},h)-\sum_{h=1}^{\infty}c^{h}P(j,S,u,h)\\
=&\sum_{h=1}^{\infty}c^{h}P^{S\cup\{u\}}(j,\{u\},h)-\sum_{h=1}^{\infty}c^{h}P^{S\cup\{u\}}(j,\{u\},h)\sum_{h=1}^{\infty}c^{h}P(u,S,h)\\
=&\sum_{h=1}^{\infty}c^{h}P^{S}(j,\{u\},h)\Big[1-\sum_{h=1}^{\infty}c^{h}P(u,S,h)\Big]\\
=&\sum_{h=1}^{\infty}c^{h}P^{S}(j,\{u\},h)\Big[1-h(u,S)\Big],
\end{aligned}$$

其中 $P^{T}(j,S,h)$ 表示从 j 出发的随机游走在第 h 步碰到 S 中点的概率且在随机游走的过程中，不经过任何 T 中的点。因此 $\Delta(u)$ 可以有如下推导过程

$$\begin{aligned}
\Delta(u)&=I(S\cup\{u\})-I(S)\\
&=1-h(u,S)+\sum_{j\in(V-S\cup\{u\})}\Big[h(j,S\cup\{u\})-h(j,S)\Big]\\
&=(1-h(u,S))\Big[1+\sum_{j\in V-S\cup\{u\}}\sum_{h=1}^{\infty}c^{h}P^{S}(j,\{u\},h)\Big].
\end{aligned}\tag{3.14}$$

因为 $0<c<1$ 且 $\sum_{h=1}^{\infty}P(u,S,h)\leqslant 1$，所以可以得到 $h(u,S)\leqslant 1$，且 $1-h(u,S)\geqslant 0$。即 $\Delta(u)\geqslant 0$，$I(S)$ 是一个非递减函数。

现在我们可以继续证明 $I(S)$ 是一个子模函数。根据子模函数的数学定义：如果对于 $S\subseteq T$，函数 $I(S)$ 满足 $I(S\cup\{u\})-I(S)\geqslant I(T\cup\{u\})-I(T)$，那么 $I(S)$ 就具有子模的性质。在证明 $I(S)$ 的非递减性时，我们可以知道当 $S\subseteq T$ 时，$P^{S}(j,\{u\},h)\geqslant P^{T}(j,\{u\},h)$。并且根据非递减特点，$h(u,S)\leqslant h(u,T)$。根据以上不等式和式(3.14)，我们可以得到当 $S\subseteq T$ 时，$I(S\cup\{u\})-I(S)\geqslant I(T\cup\{u\})-I(T)$。因此 $I(S)$ 是一个子模函数。

根据子模函数的性质，我们可以给出求 $I(S)$ 近似最优解的基本贪心算法，算法 3.2描述了基本贪心算法的运行过程。大致来讲，为了找到使得影响力中心性最大的 k 个节点的集合，算法需要运行 k 轮，并在每一轮选取出使得当前中

万方数据

```
   input : 超图 G(V, E, ℰ1, ..., ℰl) 和整数 k
   output: 由 k 个节点组成且使 I(S) 近似最大的集合 S
1  S ← ∅, I(S) ← 0;
2  for s = 1 to k do
3  |  for u ∈ (V − S) do
4  |  |  I(S ∪ {u}) ← 0;
5  |  |  for j ∈ (V − S ∪ {u}) do
6  |  |  |  I(S ∪ {u}) ← I(S ∪ {u}) + h(j, S ∪ {u});
7  |  |  end
8  |  end
9  |  v ← arg max_{u∈(V−S)} I(S ∪ {u}) − I(S);
10 |  S ← S ∪ {v};
11 end
```

算法 3.2: 最大化 I(S) 的基本贪心算法

心性增量最大的节点加入到种子集合中。该贪心算法可以保证所选取的 k 个节点组成的集合 S 和最优解 S^* 之间满足以下关系：$I(S) \geqslant (1-1/e)I(S^*)$。

前面已经分析过计算影响力中心性的时间复杂度，对于 $j \notin S$，利用蒙特卡洛算法近似计算 $h(j,S)$ 的时间复杂度为 $O(RL)$。基本贪心算法的时间复杂度为 $O(kn^2RL)$，其中 n 表示社交活动网络中的用户数。对于估计节点集合 $S \cup \{u\}$ 的影响力中心性，我们需要从所有其他节点出发总共进行 $O(nR)$ 的随机游走。因为要找出使得 I(S) 增量最大的候选节点，我们需要尝试所有候选节点 $u \in (V-S)$，因此基本贪心算法的时间复杂度为 $O(kn^2RL)$。基本贪心算法虽然将原本 NP 难的问题在多项式时间复杂度内进行求解，但对于大规模的社交网络来说，这样的时间开销仍然过大。因此我们从两方面来进一步优化贪心算法。

- **并行计算**: 贪心算法的关键步骤是计算将候选节点 u 加入到种子集合 S 后影响力中心性的增量 $\Delta(u) = I(S \cup \{u\}) - I(S)$（算法 3.2中的行 5-7）。增量 $\Delta(u)$ 可以被进一步推导为

$$\Delta(u) = \left[1 - \sum_{h=1}^{\infty} c^h P(u, S, h)\right] \times \left[1 + \sum_{j \in (V-S\cup\{u\})} \sum_{h=1}^{\infty} c^h P^S(j, \{u\}, h)\right].$$

  在基本贪心算法中，对每一个候选节点 u，我们都需要从其他所有节点 $v \in (V - S \cup \{u\})$ 出发进行 R 次随机游走。由于每一轮选出使得 $\Delta(u)$ 最大的候选节点需要遍历所有 $u \in (V-S)$，因此每选出一个种子节点总共需要 $O(n^2R)$ 次随机游走。也就是说每一轮是顺序计算所有候选节点的增量 $\Delta(u)$，这是导致基本贪心算法低效的一个重要原因。因此我们采取做法的主要思想是在每一轮同时计算出所有候选点的影响力中心性增量。具体来讲，考虑从节点 j 出发的随机游走，我们会计算将该随机游走所碰到的每一个点加入到种子集合后带来的增益。根据 $\Delta(u)$ 的表达式，我们

万方数据

可以在随机游走的过程中直接计算 $P^S(j,u,h)$。因为要从所有 $j \in (V-S)$ 出发进行随机游走，因此我们也可以得到 $P(u,S,h)$。最终在 $O(nR)$ 的随机游走结束后，我们就可以直接计算出所有候选节点给影响力中心性带来的增量。这就避免了顺序计算带来的计算开销，使时间复杂度减少为 $O(knRL)$。

- **重用随机游走**: 在通过并行计算的思想优化基本贪心算法之后，在选取每一个种子节点时，我们都需要进行 $O(nR)$ 次随机游走，因为我们的目标是选取 k 个种子节点，因此经过并行计算技术优化过后的贪心算法时间开销为 $O(knRL)$。重用随机游走的主要思想是我们只需要在选取第一个种子节点时进行 $O(nR)$ 次随机游走，并将该轮随机游走经过的点按顺序存放在内存中。当我们选取剩下的 $k-1$ 个种子节点时，我们可以利用这些已经存下来的信息进行少量的更新操作而不是重新进行随机游走。经过这步优化，我们最终可以讲时间开销减少到 $O(nRL)$。

```
input : 超图 G(V, E, ℰ_1, ..., ℰ_l) 和整数 k
output: 由 k 个节点组成且使 I(S) 近似最大的集合 S
S ← ∅, score[1...n] ← 0, P[1...n] ← 0;
for j ∈ V do
    for r = 1 to R do
        i ← j;
        visited ← ∅;
        for t = 1 to L do
            visited ← visited ∪ {i};
            i ← 根据转移概率选择下一个节点;
            RW[j][r][t] ← i;
            if i ∉ visited then
                index[i].add(item(j, r, t));
                score[i] ← score[i] + c^t/R;
            end
        end
    end
end
v ← arg max_{u∈V} score[u];
for s = 2 to k do
    update(RW, index, P, score, S, v, L);
    S ← S ∪ {v};
    v ← arg max_{u∈(V−S)} (1 − P[u])(1 + score[u]);
end
S ← S ∪ {v};
```

算法 3.3: 优化后的贪心算法

万方数据

```
1  Function Update (RW, index, P, score, S, v, L);
2  for w ∈ index[v] do
3  |  k ← L;
4  |  for t = 1 to L do
5  |  |  if RW[w.j][w.r][t] ∈ S then
6  |  |  |  k ← t;
7  |  |  |  break;
8  |  |  end
9  |  end
10 |  if k == L then
11 |  |  P[w.j] ← P[w.j] + c^t/R;
12 |  end
13 |  for i = w.t + 1 to k do
14 |  |  u ← RW[w.j][w.r][i];
15 |  |  score[u] ← score[u] − c^t/R;
16 |  end
17 end
```

算法 3.4: Update Function

以上两项优化最终将贪心算法的时间复杂度减少为 O(nRL)，其中 L 表示随机游走的最大步数，对应着我们用前 L 项来近似 $\sum_{h=1}^{\infty} c^h P^S(j,\{u\},h)$ 和 $\sum_{h=1}^{\infty} c^h P(u,S,h)$。另外为了计算 $\Delta(u)$，我们分别用 score[u] 和 P[u] 来记录 $\sum_{j\in V-S\cup\{u\}}\sum_{h=1}^{\infty} c^h P^S(j,\{u\},h)$ 和 $\sum_{h=1}^{\infty} c^h P(u,S,h)$。算法 3.3从整体上来说分为两个过程，第一个阶段（行 2-17）选出第一个种子节点，这个阶段中我们需要从所有点出发进行随机游走，并将随机游走的每一步记录下来，以便选取剩下的 k − 1 个种子节点时使用；第二个阶段是利用存储下来的随机游走的信息来进行更新操作，并选取剩下的 k − 1 个种子节点（行 18-23）。我们在算法 3.4展示了该过程涉及到的更新操作。

更新操作主要是利用第一个阶段保存的随机游走信息来更新 score 和 P，然后选取剩下的 k − 1 个种子节点。更新操作的直观解释是这样的：每当我们选出一个种子节点 v，在进行之后的随机游走时，这些随机游走碰到节点 v 时就会停止，因此每选出一个种子节点我们就需要对 score 和 P 进行更新。为了实现更新操作，对于碰到新选出的种子节点的随机游走，我们首先检查该随机游走是否已经在之前碰到过 S 中的点，并记录碰到的位置。如果之前不曾碰到 S 中的任何点，我们就可以对 P 进行更新。由于随机游走在碰到上一轮的种子节点 v 后会停止，因此对于在 v 之后被访问的节点 u，我们需要更新 score[u]。

需要指出的是经过优化的算法其空间复杂度也是 O(nRL)，这是因为我们需要对随机游走的信息进行保存。其中涉及到的数据结构主要有两个：一个是 RW，我们用它来存储 nR 个随机游走的每一步；另一个是 index，为了方便查询碰到节点 u 的随机游走，我们用 index 来对这 nR 个随机游走建立索引信息。

万方数据

3.3　本章小结

本章介绍社交活动网络模型和其上的影响力最大化算法。首先将社交活动网络抽象成超图，并将影响力传播模型扩展至其中，从而引出将用户活动考虑在内的影响力最大化问题 **IMP(SAN)**。**IMP(SAN)**涉及到计算给定节点集合的影响力和实现影响力的最大化，难以直接求解，因此本章引入影响力中心性 I(S)，将 **IMP(SAN)**转化为中心性最大化问题 **CMP**。为了解决 **CMP**，首先要解决 I(S) 的计算问题，基于随机游走，本章设计了蒙特卡洛算法来对 I(S) 进行快速计算。为了最大化 I(S)，本章首先证明 I(S) 是一个非递减的子模函数，并设计简单的贪心算法求解使 I(S) 最大的近似解。简单贪心算法的时间复杂度为 $O(kn^2RL)$，对于大规模的社交网络来说，该开销仍然过大。因此本章又从并行计算和重用随机游走两方面对简单贪心算法进行优化，最终使其复杂度减少为 $O(nRL)$。

万方数据

此页不缺内容

万方数据

第四章　实验

本章摘要： 为了展示模型和算法的有效性、效率和通用性，本章在真实的数据集上对算法进行验证，首先展示了考虑用户活动会对种子集合选取的准确性带来明显的提升，即考虑用户活动选出的种子节点在影响力延展度上要比不考虑用户活动时大很多。然后验证了算法的效率，与当前最先进的算法相比，在考虑了用户活动后，本文的算法可以在保证准确度几乎相同的情况下，大大减少算法运行时间。最后本章从模型通用性和可扩展性上进一步展示算法的效果。

4.1　数据集

我们选取了三个社交网络评分系统作为我们的数据集，分别是：Ciao[44]，Yelp[45] 和 Flixster[46]。这些数据集都是由社交网络和用户对产品的评分构成的。网络中的节点分为两类，一类代表网络中的用户，另一类是被用户评价的产品。网络中的关系也分为两类，一类是用户间的关系：好友关系对应无向边，关注关系对应有向边。另一类是用户对产品的评分，最少为 1 分，最多为 5 分。用户对产品的评价可以视为用户的活动，多个用户共同对一个产品进行评价可以视为这些用户都参加了相同的活动。我们移去了小于 3 的评分，以此保证用户对各产品的看法都是积极的。处理后我们可以认为参加同一活动的用户有着相同的兴趣爱好，这就有助于影响力通过活动在网络中传播。由于原本的 Flixster 数据集过大，导致和我们作比较的 IMM 算法 [18] 无法完成预处理，因此我们对原本的 Flixster 数据集其进行了适当的处理。由于 Flixster 中的社交网络几乎是全连通的，因此我们随机选取了一个用户，并从该用户开始进行广度优先遍历，并将遍历到的用户加入到我们要使用的数据集中，最终我们从 Flixster 中抽取了 300,000 个用户，并保留这些用户之间的关系和做出的评分，得到可以进行比较的数据集 Flixster_part。另外，我们也保留原本的 Flixster 数据集来验证我们算法的可扩展性，该数据集被称为 Flixster_entire。我们在表 4.1中列举出了这些数据集的统计信息。在实验过程中，所有的算法都运行在一台 Intel Xeon E5-2650 服务器上，处理器的主频为 2.60GHz，内存为 64GB。

Dataset	Users	Links in OSN	Products	Ratings	OSN Type
Ciao	2,342	57,544	15,783	32,783	directed
Yelp	174,100	2,576,179	56,951	958,415	directed
Flixtser_part	300,000	6,394,798	28,262	2,195,134	undirected
Flixtser_entire	786,936	11,794,648	827,798	5,372,809	undirected

表 4.1　数据集信息统计.

万方数据

4.2　考虑用户活动对影响力最大化的提升

我们首先展示考虑用户活动后，所选取的种子节点在影响力延展度上得到的提升。我们将种子节点的个数限制为 50。为了展示活动带来的影响，我们先用当前最先进的影响力最大化算法 IMM 在每个数据集对应的社交网络中选取 50 个种子用户 S_{IMM}，这 50 个用户是在不考虑用户活动的情况下选取出来的。然后我们再用我们的算法 IM-RW 在社交活动网络中选取 50 个种子用户 S_{IM-RW}，这是在考虑了用户活动后选取出来的。然后我们分别模拟计算 S_{IMM} 和 S_{IM-RW} 的影响力延展度，并分别表示为 $\sigma(\text{OSN})$ 和 $\sigma(\text{SAN})$。最后我们定义影响力延展度的提升率为

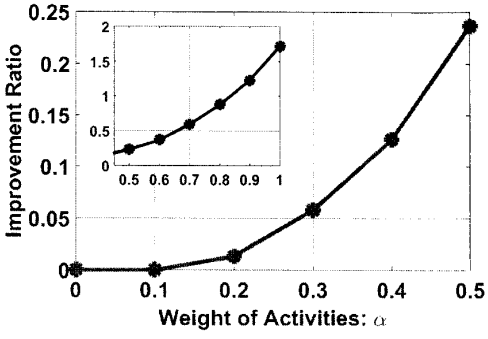
$$\text{Improvement Ratio} = [\sigma(\text{SAN}) - \sigma(\text{OSN})]/\sigma(\text{OSN}).$$

说明: IMM 算法 [18] 是目前社交网络影响力最大化问题中最先进的算法。它可以在接近线性的时间内选出种子节点集合，且对所选出来的节点正确性上有着理论保证。但 IMM 本身是针对社交网络的，即它没有考虑影响力沿着用户活动传播的情况。因此我们在此用它来作为基准算法进行比较和验证。

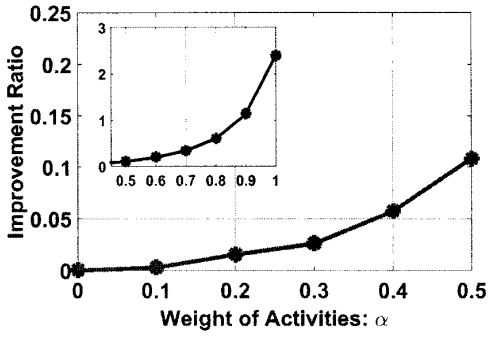
为了方便展示我们的结论，我们只考虑了网络中只有一种类型的用户和一种类型的活动时的情况，在此情况下，如果一个用户有参加活动，那么他们受到的来自于活动的影响相同。即这些用户拥有相同的活动权重 α。需要指出的是，尽管我们这里只考虑了单一用户类型和单一活动类型的情况，但我们的计算框架在多种用户类型和多种活动类型的情况下仍然可以正常使用，我们也在§4.4.2中进行了验证。

图4.1展示了将用户活动考虑进来后，不同的活动权重对影响力延展度的提升情况，其中活动权重 α 的取值范围为从 0 到 1。横坐标对应着 α 的取值，α 越大，说明活动对于影响力的传播越重要；反之，活动对影响力的传播作用越小，当 $\alpha = 0$ 时，活动对影响力的传播不起作用，此时影响力只沿着社交网络传播，问题退化成传统社交网络中的影响力传播问题。纵轴对应着影响力延展度的提升率。从图 4.1我们可以得出当 $\alpha = 0$ 时，提升率为 0。这是因为此时活动对影响力的传播不起作用，影响力只能沿着直接的用户关系进行传播。随着 α 的增加，用户活动对影响力的传播促进作用越明显，此时提升率也随之增加。这说明当网络中的用户倾向于通过活动被其他用户影响时，考虑用户活动会给种子节点的选取带来很大的增益。通过具体分析 $\alpha = 0.5$ 时的情况可以看到，对于 Ciao 数据集，其提升率在 25% 左右，也就是说考虑用户活动后，我们可以比只考虑社交网络时选出来的种子节点多影响 25% 的用户。在 Yelp 和 Flixster 这两个数据集上我们也能看到类似的结论。一个更为有趣的现象是，当 α 接近 1 时，提升率往往会成倍的增加，对于 Flixster 数据集来说，提升率甚至达到了原来的 16 倍。这些结果充分说明在解决影响力最大化问题时，将用户活动考虑在内是至关重要的。

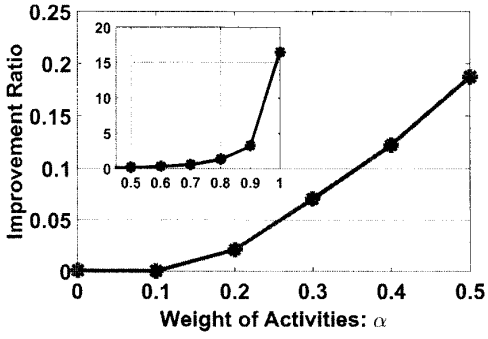
万方数据



(a) Ciao



(b) Yelp



(c) Flixster

图 4.1　独立级联模型下，考虑活动对影响力延展度的提高

4.3　算法的运行效率及准确性

在这个小节，我们通过与 IMM 算法作比较来验证我们的算法 IM-RW 的效率。我们已经提到过，IMM 是目前社交网络中最先进的影响力最大化算法，它的运行时间接近线性复杂度，且对于输出结果的准确性有着理论上的保证。IMM 本身没有考虑有用户活动存在的情况。为了比较，我们首先将包含用户活动的社交活动网络进行转换，根据 §3.1.2中对社交活动网络中两个用户间影响的刻画，我们可以将社交活动网络转换为边上有影响概率的社交网络，这时我们就可以将 IMM 算法用于考虑了用户活动的影响力最大化问题中。需要强调的是我们的算法 IM-RW 并不需要这个转换过程，我们在实验中发现从社交活动网络转换成边上有影响概率的社交网络需要耗费大量的时间，但为了比较上的公平起见，我们没有将 IMM 运行前所做预处理的消耗考虑在内。

我们首先比较了不同活动权重和不同种子集合大小下，我们的算法 IM-RW 和 IMM 的运行时间。实验结果展示在图 4.2和图 4.3中。我们可以发现，IMM 的运行时间要远远大于我们的算法 IM-RW，特别是在网络规模增大和用户活动权重 α 增大的情况下。这对应着以下解释：虽然 IMM 的运行时间接近线性的复杂度，但它是和网络中的边数相关的，当我们将考虑了用户活动的社交活动网

万方数据

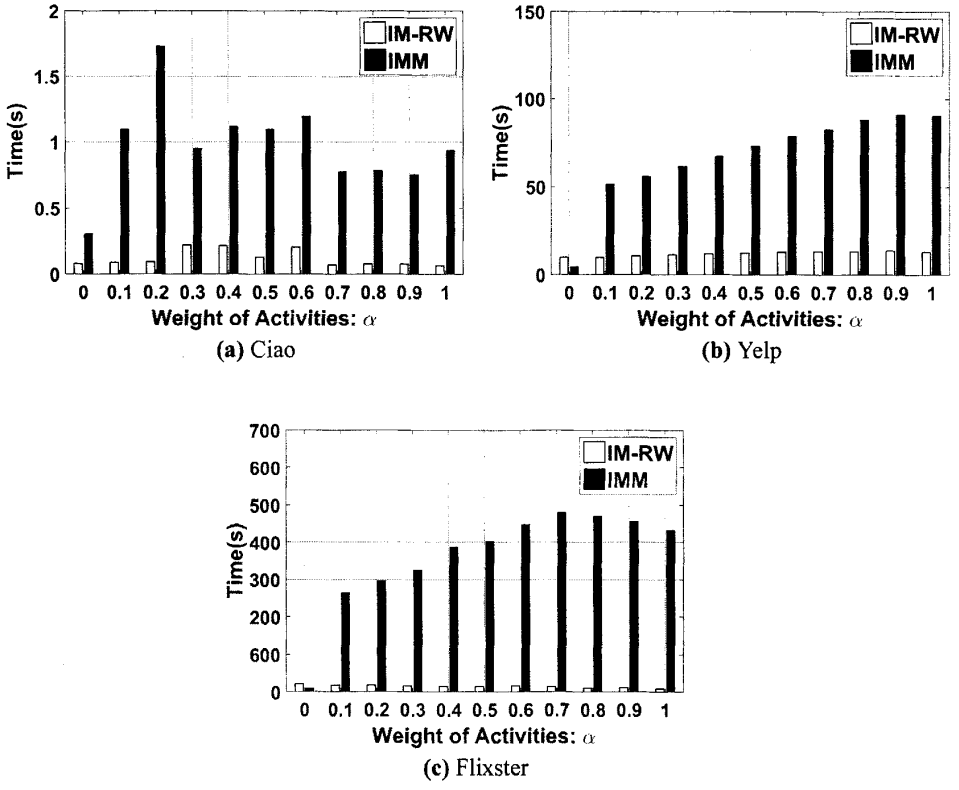


图 4.2　不同活动权重，IMM 和 IM-RW 算法的运行时间

络转换为带有影响概率的网络后，用户活动会使原本没有连接的用户建立联系，所以转换后的网络会变得非常稠密，这就导致了 IMM 的时间开销增大。通过§3.2.4中的分析可以发现，我们的算法 IM-RW 的时间复杂度为 O(nRL)，也就是说我们的算法中时间复杂度只和网络中的用户数有关，因此考虑用户活动也不会对我们的时间开销有太大影响。我们也展示了将 α 固定为 0.8 时，IM-RW 和 IMM 在不同种子集合大小下的运行时间，我们依旧可以发现，不论最终要选取的种子节点是多少，我们的算法 IM-RW 在运行时间上都要远小于 IMM，且我们的算法在时间开销上是独立于需要选取的种子节点数的，这得益于我们对基本贪心算法所做的第二步优化。通过这组实验我们可以得出这样的结论：我们的算法 IM-RW 在解决考虑了用户活动的影响力最大化问题时，可以大大地提高算法的运行效率。

接下来我们展示我们算法所选出来的种子节点的影响力延展度，并与 IMM 作比较。比较结果展示在图4.4中。横坐标对应着活动权重 α，纵坐标对应模拟计算的影响力延展度。对影响力延展度的计算是在转换得到的网络中，按照之前介绍的独立级联模型的工作流程进行 1000 次模拟计算，并将这 1000 次计算的结果取平均值作为最终的影响力延展度。我们可以发现，IM-RW 和 IMM 在

万方数据

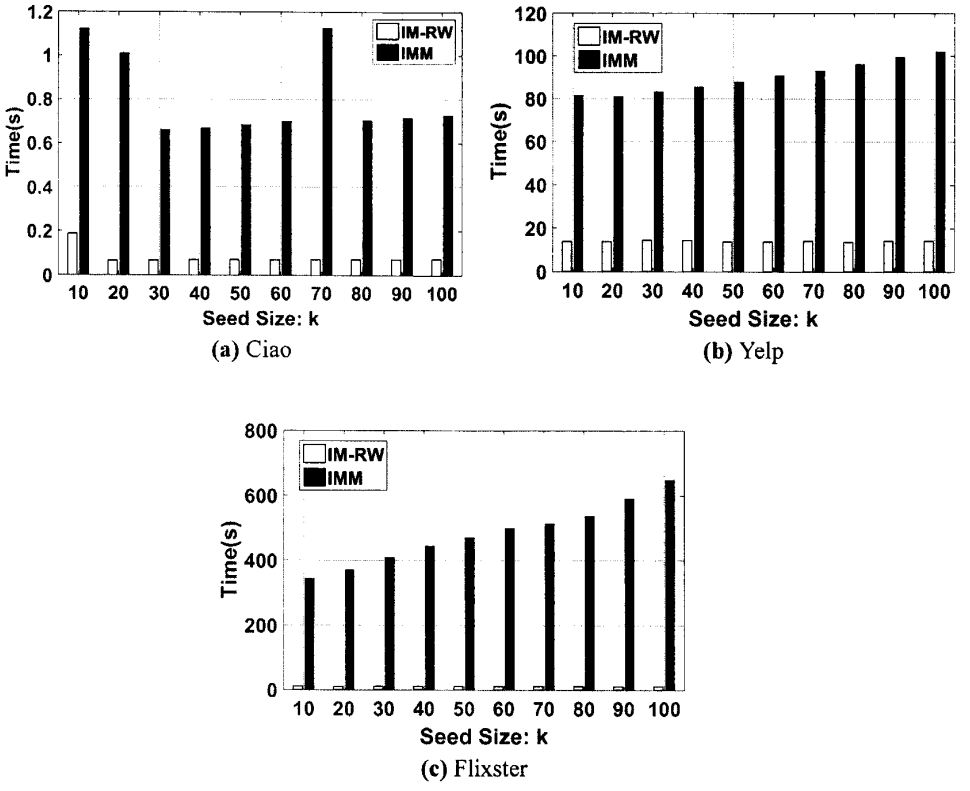


图 4.3　不同种子集合大小，IMM 和 IM-RW 算法的运行时间

影响力延展度上几乎一样。

总结: 对于考虑了用户活动的影响力最大化问题，我们的 IM-RW 算法可以在时间上和准确性上得到很好的性能。通过与当前最先进的算法 IMM 的比较，可以发现在保证几乎相同准确性的情况下，我们算法的时间开销要远远小于 IMM。

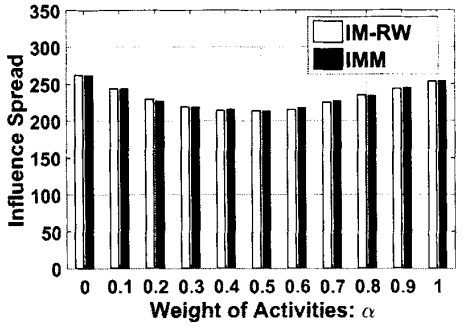
4.4　模型通用性

我们从两方面验证模型的通用性：一是将独立级联模型替换为线性阈值模型，二是考虑有多种用户和活动存在的异构网络。我们会从将活动考虑进来以后的提升率和 IM-RW 算法的运行时间两个方面来展示模型地通用性。

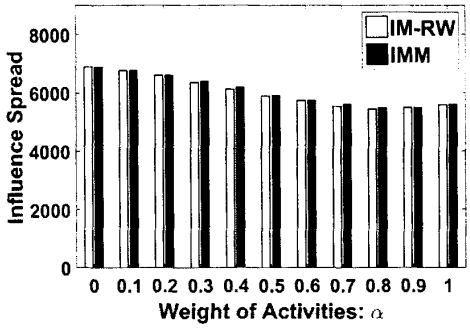
4.4.1　线性阈值模型

我们发现，除了独立级联模型，我们的算法也可以用于线性阈值模型。此时式 (3.1) 中所示的用户间的影响就对应着线性阈值模型中边上的权重。我们仍考虑一种用户和一类活动的简单情形。我们首先在图4.5 中展示了将用户活动考虑在内后，所选出的种子集合在线性阈值模型下影响力延展度的提升百分比。我们发现，同在独立级联模型下一样，影响力延展度有着很大的提高，这说明

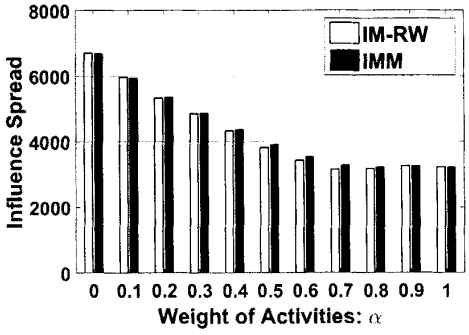
万方数据



(a) Ciao



(b) Yelp



(c) Flixster

图 4.4　独立级联模型下，IMM 和 IM-RW 的影响力延展度

在不同的影响力传播模型中，将活动考虑在内对于准确选取影响力最大的节点集合有着重要作用。另外，如表 4.2中所示，我们也给出了我们的算法的运行时间，对于 Ciao 数据集，算法的运行时间在 0.1 秒左右，对于 Yelp 和 Flixster 数据集则分别在 10 秒和 20 秒左右。可以看出来我们的算法对于线性阈值模型来说依旧有着很高的运行效率。

数据集	运行时间 (s)										
	$\alpha = 0$	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Ciao	0.105	0.114	0.100	0.103	0.115	0.097	0.102	0.118	0.117	0.100	0.104
Yelp	8.396	9.838	10.655	11.267	11.940	12.321	12.828	13.362	13.382	13.639	14.147
Flixster	18.169	18.233	17.448	21.350	15.352	14.618	12.965	11.833	10.717	8.575	6.121

表 4.2　线性阈值模型下，IM-RW 算法的运行时间.

4.4.2　多种用户和活动类型

为了研究异构网络中算法的工作情况，我们考虑了三种设置：首先是用户异构，我们将用户分为两种；然后是活动异构，我们将活动分为两类；最后是全异构，我们将用户和活动分别设置为两类。

由于数据集中没有给出用户和活动的分类信息，我们根据 2-8 原则 [47] 人

万方数据

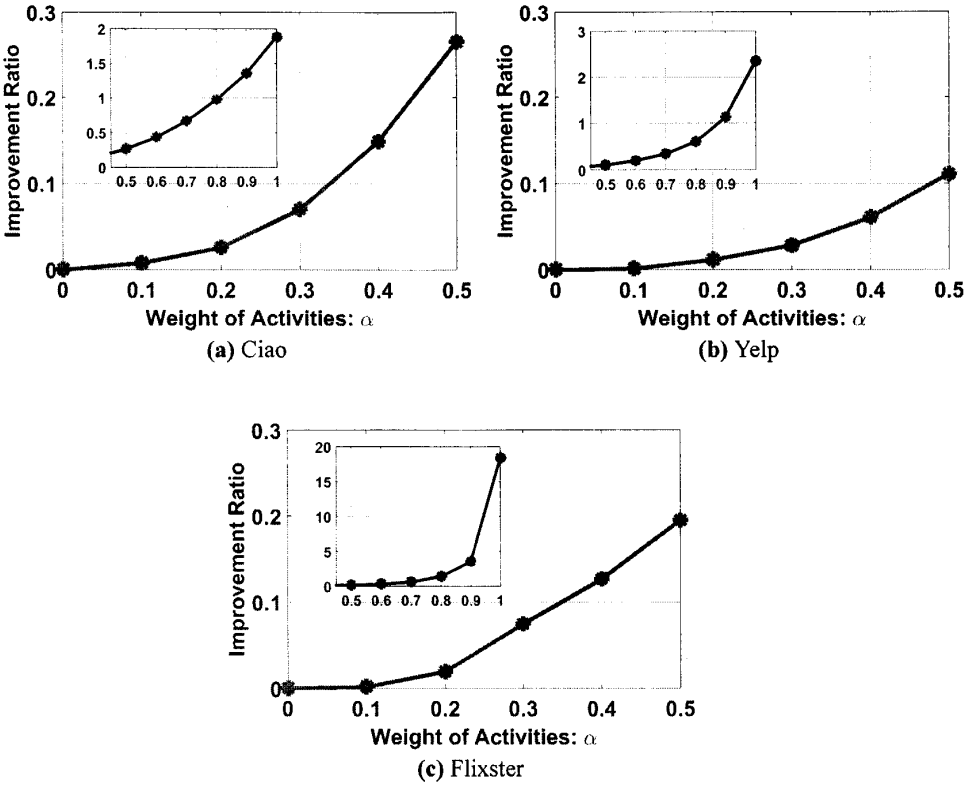


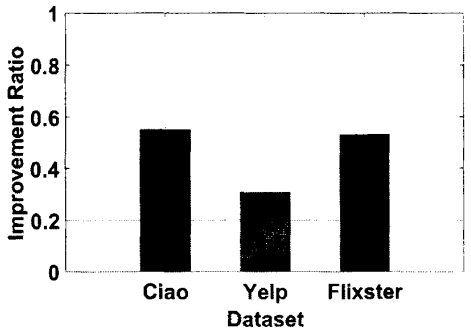
图 4.5　线性阈值模型下，考虑活动对影响力延展度的提高

工对用户和活动进行分类，该原则广泛使用在计算机研究的各个领域。具体来说，在考虑用户异构时，我们随机的将用户分为两种，其中第一种占 80%，第二种占 20%，由于我们更关心用户偏向于受活动影响的情况，因此将第一类用户被活动影响的权重设置为 0.8，将第二类用户被活动影响的权重设置为 0.2。在考虑活动异构时，我们将用户受到活动影响的权重统一设置为 0.8，然后利用 2-8 原则将活动分为两类，第一类活动占 80%，其权重设置为 $0.8 \times 0.8 = 0.64$；第二类活动占 20%，其权重设置为 $0.8 \times 0.2 = 0.64$。对于全异构的情况，我们综合前两种的设置，第一类用户受活动影响的权重设置为 0.8，其中第一类活动的权重设置为 $0.8 \times 0.8 = 0.64$，第二类设置为 $0.8 \times 0.2 = 0.64$；第二类用户受活动影响的权重设置为 0.2，其中第一类活动的权重设置为 $0.2 \times 0.8 = 0.16$，第二类设置为 $0.2 \times 0.2 = 0.04$。

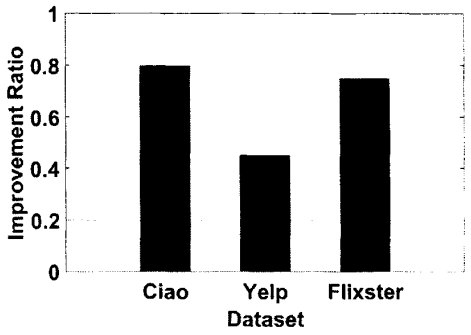
数据集	用户异构	活动异构	全异构
Ciao	0.06	0.08	0.06
Yelp	13.25	8.61	9.02
Flixtser	11.87	8.99	10.18

表 4.3　多种用户和活动类型下，1M-RW 算法的运行时间.

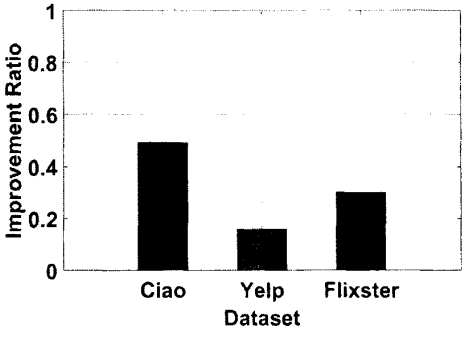
万方数据



(a) 用户异构



(b) 活动异构



(c) 全异构

图 4.6　多种用户和活动类型下，考虑活动对影响力延展度的提高

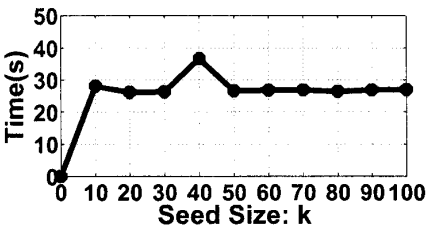
图 4.6展示了在以上所述异构设置下将用户活动考虑在内后对影响力延展度的提升情况。我们发现不管在哪种情况下，提高率都达到了 18% 以上，对于活动异构的情形，Ciao 和 Flixster 上的提高率甚至达到了 70% 以上。这说明将用户考虑在内后，即使在异构网络中，也会大大提高种子节点的准确度。另外，我们也在表 4.3中展示了在不同设置中我们的算法 IM-RW 的运行时间，可以发现 Ciao 数据集上的运行时间在 0.1 秒以内，而 Yelp 和 Flixster 数据集上的其运行时间也只是在 15 秒以内，这说明即使在异构情况下，我们的算法仍有很高的运行效率。

4.5　模型可扩展性

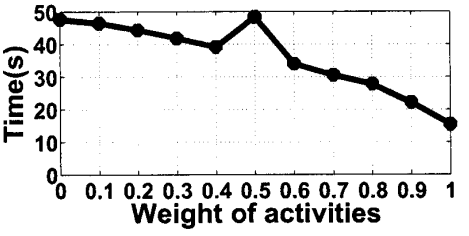
为了评估我们算法在更大的图上的运行时间，我们在完整的 Flixster 数据集上运行我们的算法 IM-RW。需要指出的是 IMM 算法运行完整的 Flixster 数据集需要有一个预处理过程，由于该预处理过程的时空开销很大，因此在我们的实验环境下无法完成。我们的 IM-RW 算法不需要进行该预处理，因此依旧可以运行。我们在图4.7中展示了我们算法在不同种子集合大小和不同活动权重下的运行时间，可以发现算法都可以在 50 秒内完成。并且得益于我们对基本贪心算法

万方数据

进行的第二个优化，算法的运行时间独立于种子集合的大小；另外，可以看出活动权重的增大也不会增加我们算法的开销。这些都说明我们的算法可以扩展到更大规模的网络中去。



(a) 不同种子集合大小，IM-RW 的运行时间



(b) 不同活动权重，IM-RW 的运行时间

图 4.7　完整 Flixster 数据集下，IM-RW 算法的运行时间.

4.6　本章小结

本章在真实的数据集上验证模型和算法的有效性、效率和通用性。首先展示了考虑用户活动对种子集合选取的影响。从实验结果可以发现，考虑用户活动会显著提升所选取种子节点的影响力延展度：当将活动权重固定为 0.5 时，在数据集 Ciao 上的提升率可以达到 25%，而在 Yelp 和 Flixster 数据集上也分别达到了 10% 和 20%；当活动权重接近 1 时，提升率甚至可达数倍。从算法的运行时间来看，在考虑用户活动后，本文算法在保证影响力延展度几乎相同的情况下，运行速度会比当前最优算法快几倍甚至几十倍。本章还展示了算法的通用性，在线性阈值模型下，本文算法在准确性和运行效率上仍然可以取得与独立级联模型下类似的效果；在有多种用户和活动存在的异构网络中，考虑用户活动也会对种子集合选取的准确率有很大提高，且算法在不同数据集上都可在 0.1 秒到 15 秒之内完成。本章最后展示了在完整的 Flixster 数据集上，本文算法在 50 秒内就可以运行完成，且运行时间独立于种子集合的大小，也不会随着活动权重的增大而提升，说明算法具有一定的可扩展性。

万方数据

此页不缺内容

万方数据

第五章　总结与展望

5.1　总结

本文研究社交活动网络中的影响力最大化问题。影响力最大化问题是伴随着社交网络的流行而出现的。其应用包括公司可以通过在社交网络中选取影响力大的用户来帮助其推广产品，以使得该产品在人群中大范围普及等。传统的影响力最大化问题只考虑直接的用户关系而忽略了由用户活动产生的间接联系，并且存在随着网络规模增加时间开销过大的问题。这就促使我们将用户活动考虑在内来重新研究影响力最大化问题。由于单单考虑直接关系本身已经是 NP 难题，因此将庞大的用户活动考虑在内后，影响力最大化问题会变得更加难以求解。

为此，本文首先将包含用户活动的社交网络抽象为超图模型。超图中的点代表网络中的用户，超图中的边分为两大类，一类表示用户间的直接关系，一类表示用户活动产生的间接关系。然后将独立级联模型扩展至该超图模型中，定义出社交活动网络中的影响力最大化问题。为了解决计算开销过大的问题，本文利用超图上的随机游走来近似影响力在社交活动网络中的传播过程，并定义了反应影响力的中心性指标。利用蒙特卡洛框架，本文提出了可以快速计算该中心性的近似算法，并利用子模函数的性质设计了实现中心性最大化的贪心算法。基本贪心算法涉及到对中心性指标的多次计算，时间开销依然很大，本文又从两个方面对其进行优化：一是利用并行计算的思想，在随机游走的同时计算将碰到的节点加入到种子集合所带来的边际增益；二是利用空间换时间的思想，只在选取第一个种子节点时进行随机游走，并将这些随机游走的路径记录下来以便选取其他种子节点时重复使用。

通过实验可以验证：一，在解决影响力最大化问题时，将用户活动考虑在内会使选出来的种子节点的准确率大大提高：当将活动权重固定为 0.5 时，在数据集 Ciao 上的提升率可以达到 25%，而在 Yelp 和 Flixster 数据集上也分别达到了 10% 和 20%；当活动权重接近 1 时，提升率甚至可达数倍。二，当将用户活动考虑在内后，本文算法相比于当前最优的算法，在保证影响力延展度几乎相同的情况下，运行速度会提高几倍甚至几十倍。三，线性阈值模型下，本文算法仍能取得与独立级联模型下类似的性能；在多种用户和活动存在的情况下，本文算法也会对种子节点选取的准确率有很大提高，且有着很高的运行效率。四，对于当前最优算法无法处理的数据集 Flixster_entire，本文算法在 50 秒内就可以完成，且运行时间独立于种子集合的大小，也不会随着活动权重的增大而提升。

万方数据

5.2 展望

尽管本文的计算框架可以快速有效处理将用户活动考虑在内的影响力最大化问题，但依然可以从以下几方面继续对该问题的研究：

1. 本文的研究工作基于图可以被完全放入内存的前提。当今的社交网络其规模已经大到无法直接放入内存处理，为此有许多基于外存来处理大图的单机系统被开发出来，比如 GraphChi，Xstream 和 FlashGraph 等，从实验可以看出，这些系统已经可以在单机上处理有几十亿条边的大图。如何利用这些大规模的图处理系统来进行有效的随机游走，解决影响力最大化问题有待我们进一步研究。

2. 本文主要考虑均匀的权重设置，比如一个用户被其直接邻居影响的权重是相同的。虽然我们的计算框架可以直接扩展至任意权重设置的情况，但直接扩展后的计算效率会受到一定影响，如何在保证计算效率不受过多影响的前提下处理任意的权重设置是我们面临的又一个难题。

3. 本文中权重是作为算法的输入来处理的，实际应用中需要先获得权重。由于考虑用户活动后参数会变得更多，因此需要更有效的权重学习算法。

万方数据

参考文献

[1] Facebook. https://investor.fb.com.
[2] Sina Weibo. http://tech.sina.com.cn.
[3] 失恋 33 天. https://en.wikipedia.org/wiki/Love_Is_Not_Blind.
[4] 冰桶挑战. https://en.wikipedia.org/wiki/Ice_Bucket_Challenge.
[5] Christakis N A, Fowler J H. The Spread of Obesity in a Large Social Network over 32 Years. New England Journal of Medicine, 2007, 357(4):370–379.
[6] Christakis N A, Fowler J H. The Collective Dynamics of Smoking in a Large Social Network. New England Journal of Medicine, 2008, 358(21):2249–2258.
[7] Aral S, Walker D. Identifying Influential and Susceptible Members of Social Networks. Science, 2012, 337(6092):337–341.
[8] Bond R M, Fariss C J, Jones J J, et al. A 61-Million-Person Experiment in Social Influence and Political Mobilization. Nature, 2012, 489(7415):295–298.
[9] Kempe D, Kleinberg J, Tardos É. Maximizing the Spread of Influence Through a Social Network. Proceedings of the 9th International Conference on Knowledge Discovery and Data Mining. ACM, 2003. 137–146.
[10] Granovetter M. Threshold Models of Collective Behavior. American Journal of Sociology, 1978, 83(6):1420–1443.
[11] Liggett T. Interacting Particle Systems, volume 276. Springer Science & Business Media, 2012.
[12] Netrapalli P, Sanghavi S. Learning the Graph of Epidemic Cascades. Proceedings of the 42nd International Conference on Measurement and Modeling of Computer Systems, volume 40. ACM, 2012. 211–222.
[13] Saito K, Nakano R, Kimura M. Prediction of Information Diffusion Probabilities for Independent Cascade Model. Proceedings of the 14th International Conference on Knowledge-Based Intelligent Information and Engineering Systems. Springer, 2008. 67–75.
[14] Goyal A, Bonchi F, Lakshmanan L V. Learning Influence Probabilities in Social Networks. Proceedings of the 3rd International Conference on Web Search and Data Mining. ACM, 2010. 241–250.
[15] Karp R M. Reducibility among Combinatorial Problems. Proceedings of Complexity of Computer Computations. Springer, 1972: 85–103.
[16] Nemhauser G L, Wolsey L A, Fisher M L. An Analysis of Approximations for Maximizing Submodular Set Functions—I. Mathematical Programming, 1978, 14(1):265–294.
[17] Zhao P, Li Y, Xie H, et al. Measuring and Maximizing Influence via Random Walk in Social Activity Networks. Proceedings of the 22nd International Conference on Database Systems for Advanced Applications. Springer, 2017. 323–338.
[18] Tang Y, Shi Y, Xiao X. Influence Maximization in Near-Linear Time: A Martingale Approach. Proceedings of the 36th International Conference on Management of Data. ACM, 2015. 1539–1554.
[19] Freeman L C. Centrality in Social Networks Conceptual Clarification. Social Networks, 1978, 1(3):215–239.
[20] Olsen P W, Labouseur A G, Hwang J H. Efficient Top-k Closeness Centrality Search. Proceedings of the 30th International Conference on Data Engineering. IEEE, 2014. 196–207.
[21] Zhao J, Lui J, Towsley D, et al. Measuring and Maximizing Group Closeness Centrality over Disk-Resident Graphs. Proceedings of the 23rd International Conference on World Wide Web. ACM, 2014. 689–694.
[22] Brandes U, Pich C. Centrality Estimation in Large Networks. International Journal of Bifurcation and Chaos, 2007, 17(07):2303–2318.
[23] Riondato M, Kornaropoulos E M. Fast Approximation of Betweenness Centrality through Sampling. Data Mining and Knowledge Discovery, 2016, 30(2):438–475.
[24] Page L, Brin S, Motwani R, et al. The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Stanford InfoLab, 1999.
[25] Hochbaum D S. Approximation Algorithms for NP-hard Problems. PWS Publishing Co., 1996.
[26] Valiant L G. The Complexity of Computing the Permanent. Theoretical Computer Science, 1979, 8(2):189–201.

万方数据

[27] Chen W, Wang Y, Yang S. Efficient Influence Maximization in Social Networks. Proceedings of the 15th International Conference on Knowledge Discovery and Data Mining. ACM, 2009. 199–208.
[28] Leskovec J, Krause A, Guestrin C, et al. Cost-Effective Outbreak Detection in Networks. Proceedings of the 13th International Conference on Knowledge Discovery and Data Mining. ACM, 2007. 420–429.
[29] Goyal A, Lu W, Lakshmanan L V. Celf++: Optimizing the Greedy Algorithm for Influence Maximization in Social Networks. Proceedings of the 20th International Conference on World Wide Web. ACM, 2011. 47–48.
[30] Wang C, Chen W, Wang Y. Scalable Influence Maximization for Independent Cascade Model in Large-Scale Social Networks. Data Mining and Knowledge Discovery, 2012, 25(3):545.
[31] Goyal A, Lu W, Lakshmanan L V. Simpath: An Efficient Algorithm for Influence Maximization under the Linear Threshold Model. Proceedings of the 11th International Conference on Data Mining. IEEE, 2011. 211–220.
[32] Borgs C, Brautbar M, Chayes J, et al. Maximizing Social Influence in Nearly Optimal Time. Proceedings of the 25th ACM-SIAM Symposium on Discrete Algorithms. SIAM, 2014. 946–957.
[33] Tang Y, Xiao X, Shi Y. Influence Maximization: Near-Optimal Time Complexity Meets Practical Efficiency. Proceedings of the 35th International Conference on Management of Data. ACM, 2014. 75–86.
[34] Jeh G, Widom J. SimRank: A Measure of Structural-Context Similarity. Proceedings of the 8th International Conference on Knowledge Discovery and Data Mining. ACM, 2002. 538–543.
[35] Kusumoto M, Maehara T, Kawarabayashi K. Scalable Similarity Search for SimRank. Proceedings of the 35th International Conference on Management of Data. ACM, 2014. 325–336.
[36] Salganik M J, Heckathorn D D. Sampling and Estimation in Hidden Populations Using Respondent-Driven Sampling. Sociological Methodology, 2004, 34(1):193–240.
[37] Stutzbach D, Rejaie R, Duffield N, et al. On Unbiased Sampling for Unstructured Peer-to-Peer Networks. Transactions on Networking, 2009, 17(2):377–390.
[38] Avrachenkov K, Ribeiro B, Towsley D. Improving Random Walk Estimation Accuracy with Uniform Restarts. Proceedings of International Workshop on Algorithms and Models for the Web-Graph. Springer, 2010. 98–109.
[39] Shang S, Kulkarni S R, Cuff P W, et al. A Random Walk Based Model Incorporating Social Information for Recommendations. Proceedings of International Workshop on Machine Learning for Signal Processing. IEEE, 2012. 1–6.
[40] Yin H, Cui B, Li J, et al. Challenging the Long Tail Recommendation. Proceedings of the 38th International Conference on Very Large Databases. Morgan Kaufmann & ACM, 2012. 896–907.
[41] Bellaachia A, Al-Dhelaan M. Random Walks in Hypergraph. Proceedings of International Conference on Applied Mathematics and Computational Methods, 2013. 187–194.
[42] Chen W, Wang C, Wang Y. Scalable Influence Maximization for Prevalent Viral Marketing in Large-Scale Social Networks. Proceedings of the 16th International Conference on Knowledge Discovery and Data Mining. ACM, 2010. 1029–1038.
[43] Hoeffding W. Probability Inequalities for Sums of Bounded Random Variables. Journal of the American Statistical Association, 1963, 58(301):13–30.
[44] Tang J, Gao H, Liu H. mTrust: Discerning Multi-Faceted Trust in a Connected World. Proceedings of the 5th International Conference on Web Search and Data Mining. ACM, 2012. 93–102.
[45] Yelp dataset. https://www.yelp.com/dataset_challenge/dataset.
[46] Jamali M, Ester M. A Matrix Factorization Technique with Trust Propagation for Recommendation in Social Networks. Proceedings of the 4th Conference on Recommender Systems. ACM, 2010. 135–142.
[47] Trueswell R L. Some Behavioral Patterns of Library Users: The 80/20 Rule. Wilson Libr Bull, 1969..
[48] Domingos P, Richardson M. Mining the Network Value of Customers. Proceedings of the 7th International Conference on Knowledge Discovery and Data Mining. ACM, 2001. 57–66.
[49] Chen W, Lu W, Zhang N. Time-Critical Influence Maximization in Social Networks with Time-delayed Diffusion Process. Proceedings of the 26th Conference on Artificial Intelligence, 2012. 1–5.

万方数据

[50] Centola D, Macy M. Complex Contagions and the Weakness of Long Ties 1. American Journal of Sociology, 2007, 113(3):702–734.
[51] Chen W, Lakshmanan L V, Castillo C. Information and Influence Propagation in Social Networks. Synthesis Lectures on Data Management, 2013, 5(4):1–177.
[52] Anagnostopoulos A, Kumar R, Mahdian M. Influence and Correlation in Social Networks. Proceedings of the 14th International Conference on Knowledge Discovery and Data Mining. ACM, 2008. 7–15.
[53] Barbieri N, Bonchi F, Manco G. Topic-Aware Social Influence Propagation Models. Knowledge and Information Systems, 2013, 37(3):555–584.
[54] Lu W, Chen W, Lakshmanan L V. From Competition to Complementarity: Comparative Influence Diffusion and Maximization. Proceedings of the 41th International Conference on Very Large Databases, 2015. 60–71.
[55] McPherson M, Smith-Lovin L, Cook J M. Birds of a Feather: Homophily in Social Networks. Annual Review of Sociology, 2001, 27(1):415–444.
[56] 陈卫. 社交网络影响力传播研究. 大数据, 2017, 1(3):2015031.

万方数据

此页不缺内容

万方数据

致 谢

一转眼，三年的研究生生涯即将结束，刚入学时的场景还历历在目。在研究生期间的学习生活，将使我终身受益。在此，我真诚地感谢研究生期间帮助过我的老师和同学，是你们的帮助让我克服一个个困难，解决一个个问题，你们的帮助让我成长。

首先感谢我的导师李永坤副教授。李老师不仅专业知识丰富，而且年轻充满活力，对我的指导更是做到了言传身教，他几乎每周都会了解我的工作进展，一起讨论解决我遇到的困难，能在繁忙的工作中抽出固定的时间放在学生培养身上让我非常感激。李老师在带领我做研究课题的同时也给了我足够的自由度，他启发我如何思考问题，解决问题，真的做到了授之以渔。李老师对我的帮助不仅仅是在专业知识上，他努力工作仍能兼顾家庭，严于律己，宽以待人，这对我以后的工作生活将是一笔宝贵的财富。

我还要感谢许胤龙教授。许老师知识渊博，治学严谨，在这三年的时间里我目睹了许老师风雨无阻的工作状态，他让我认识到无论身居何位都要勤勉刻苦。许老师为人朴素，淡泊名利，更是我以后工作生活的榜样。

感谢实验室的同学特别是潘玉彪、梁杰、王昕、邹懋、李志鹏、张伟韬、沈标标师兄对我科研和生活上的帮助。我还要感谢同届的陈友旭、郭帆、魏舒展、孙东东、巨高莹，与他们一起做科研是一件快乐的事情。感谢其他的学弟学妹们，他们是吴志勇、张月明、赵倩倩、杨陈、汪睿，希望你们学习进步，一切顺利。

感谢寝室的三位小伙伴：韩旭、周锋、刘正夫，跟你们一起多了很多欢乐。还要感谢陪伴多年的好友马鹏、贺强，愿友谊天长地久。

感谢我最亲爱的父母，感谢你们对我二十多年来的养育之恩，感谢你们对我毫不吝啬的关爱，感谢你们无怨无悔的付出，感谢你们对我一切决定的理解和支持，是你们让我拥有克服困难的勇气与决心，你们永远是我最温暖的依靠。感谢我的妹妹，她努力认真，积极向上的工作和生活态度给了我很多激励。感谢我的女朋友郑钰，还好遇到你，你的陪伴会是我一直努力的动力。再次感谢我的家人们，希望你们身体健康，永远幸福！

最后，向评审论文的老师和专家表示深深的感谢！

赵鹏鹏

2017 年 5 月 23 日

万方数据

此页不缺内容

万方数据

在读期间发表的学术论文与取得的研究成果

已发表论文：

1. **Pengpeng Zhao**, Yongkun Li, Hong Xie, Zhiyong Wu, Yinlong Xu , John C. S. Lui. "Measuring and Maximizing Influence via Random Walk in Social Activity Networks", International Conference on Database Systems for Advanced Applications(DASFAA'2017). Springer, Cham, 2017: 323-338.

万方数据