武漢大學

本科毕业论文（设计）

IC 传播机制下的双层网络影响力最大模型

姓　　名:　王　翔

学　　号:　2020302011088

专　　业:　数据科学与大数据技术

学　　院:　数学与统计学院

指导教师:　胡　捷

二○二五年五月

原创性声明

本人郑重声明：所呈交的论文（设计），是本人在指导教师的指导下，严格按照学校和学院有关规定完成的。除文中已经标明引用的内容外，本论文（设计）不包含任何其他个人或集体已发表及撰写的研究成果。对本论文（设计）做出贡献的个人和集体，均已在文中以明确方式标明。本人承诺在论文（设计）工作过程中没有伪造数据等行为。若在本论文（设计）中有侵犯任何方面知识产权的行为，由本人承担相应的法律责任。

作者签名：　　指导教师签名：

日　　期：　　年　　月　　日

版权使用授权书

本人完全了解武汉大学有权保留并向有关部门或机构送交本论文（设计）的复印件和电子版，允许本论文（设计）被查阅和借阅。本人授权武汉大学将本论文的全部或部分内容编入有关数据进行检索和传播，可以采用影印、缩印或扫描等复制手段保存和汇编本论文（设计）。

作者签名：　　指导教师签名：

日　　期：　　年　　月　　日

摘　　要

在社交网络中，人们可以更加便捷的进行信息的获取与传播，随着科技的发展与在线社交技术的成熟，信息传播也变得更加迅速与广泛，并逐渐发展成传统线下社交网络和线上社交网络并行的模式。因此过去基于单层网络影响力最大化问题的研究已经不能很好的起到作用，需要深入对双层网络影响力最大化问题的研究。在实际生活中我们不难发现，线下的社交网络与线上的社交网络有着很大的不同，线上的社交网络的范围更加广，关系紧密度也更加弱，且往往是建立在线下的社交网络基础上发展起来的。因此，建立起恰当的双层网络模型是十分重要的。

为了更好模拟现实里的双层社交网络，本文提出了强、弱关系网络构建方法，并将它们组合构成双层社交网络; 为了更符合现代社会信息传播迅速且密集的特点，贴合实际生活（即往往会有多个来源为你提供同一件事情的信息），采用独立级联传播模型来进行实验；为了衡量节点的影响力大小，本文采用度中心性和介数中心性两个指标进行混合度量节点影响力；本文进行数值实验，从而验证了设计的方法在解决双层网络上的影响力最大化问题的可行性与有效性。

关键词：社交网络；双层网络；独立级联模型；影响力最大化

ABSTRACT

In social networks, people can obtain and disseminate information more conveniently. With the development of science and technology and the maturity of online social technology, information dissemination has become faster and more extensive, and has gradually developed into traditional offline social networks and online social networks combined model on social networks. Therefore, past research based on the problem of maximizing the influence of a single-layer network can no longer play a good role, and it is necessary to conduct in-depth research on the problem of maximizing the influence of a double-layer network. In real life, it is not difficult to find that offline social networks are very different from online social networks. Online social networks have a wider scope and weaker relationships, and are often established based on offline network, developing on the basis of offline social networks. Therefore, it is very important to establish an appropriate two-layer network model.

In order to better simulate the two-layer social network in reality, this article proposes a method for constructing strong and weak relationship networks, and combines them to form a two-layer social network; in order to be more in line with the rapid and intensive characteristics of modern society's information dissemination, it is more suitable for real life. (That is, there are often multiple sources providing you with information about the same thing). An independent cascade propagation model is used to conduct experiments; in order to measure the influence of nodes, this article uses two indicators: degree centrality and betweenness centrality.Hybrid measurement of node influence; this paper conducts numerical experiments to verify the feasibility and effectiveness of the designed method in solving the problem of maximizing influence on a two-layer network.

Key words: Social network; Double layer network; Independent cascade model; Maximizing influence

目　录

1 绪论

1.1 研究背景

社交网络中，人们可以便捷的进行信息的获取与传播，随着科技的发展与在线社交技术的成熟，信息传播变得更加迅速与广泛。为了优化和控制信息的传播，人们提出了“影响力最大化问题”，致力于在社交网络中求得一组初始节点，希望通过这些初始节点传递信息可以使得信息传播的范围尽可能地大。

在影响力最大化这一大类问题内，不同问题背景下所需要的模型也是不尽相同的，需要根据具体的问题来建立恰当的网络模型。每个具体的问题都有自己的侧重点，例如在疫情期间，一则“双黄连口服液可以抑制新型冠状病毒”的谣言经部分媒体报道后迅速传播，引发民众深夜抢购，造成民众恐慌等等问题，此时需要控制住谣言传播的范围；相反，在官方发布政策需要配合时，则要尽可能的扩大信息扩散的范围。这两个看上去方向完全相反的问题，其实本质都是对影响力最大化问题的运用。

随着大数据时代的到来，社交网络数据的规模与复杂性来到了前所未有的高度，大大提高了分析与建模社交网络影响力最大化问题的难度。

对于影响力最大化问题，经典的解决办法是运用独立级联（Independent Cascade, IC）模型或线性阈值（Linear Threshold Model, LT）模型这两种信息传播模型的贪心算法，并且获得的优化近似解能够达到 $(1-1/e-\varepsilon)$[1] 的近似比；后续提出的算法大致可以分为三类：基于仿真的方法、基于代理（proxy）的方法、基于草图的方法。

线性阈值模型指在信息传播的过程中，每个节点都有一个自己被激活的阈值，例如果某个节点的处于激活态的邻居节点的激活概率按照权重相加后大于这个节点的激活阈值，那么这个节点就会被激活；独立级联模型指在信息传播过程中，每个处于激活态的节点都进行一次以相应概率激活自己的处于非激活态的邻居节点的尝试。

随着线上社交媒体的发展，现在的信息传播逐渐发展出传统线下社交网络和线上社交网络并行的模式，基于单层网络影响力最大化问题的理论已经不能很好的应用在现实网络中，因此需要深入对双层网络影响力最大化问题的研究。

线上社交网络与传统的线下社交网络有着相似的部分，因此在构建这两种网络时可以借鉴以往构建单层社交网络的经验；但不可否认的是，线下社交网络往

往是较为“强”的关系，有关系的双方往往在例如血缘、职业、经济等多个社会经济角度的交叉比较深，在线下社交网络中信息传播的概率相对较大，体现为父母、同学、同事分享的信息人们会更倾向于相信并接受；而线上社交网络往往是较为“弱”的关系，有关系的双方往往只需要有一个共同爱好、共同话题等就可以在网络上找到一个聚集地从而进行沟通交流，交流之后通常不会维持与同一个人的交流关系，这种了解不深的关系不利于信息的传播，例如刚认识的网友分享的信息人们更倾向于持有怀疑态度；线上社交媒体还会根据用户的粉丝数来进行信息推送，从而影响其他用户的信息获取情况，体现为粉丝数量多的博主发的新闻会让更多人接收到，而粉丝数量少的博主发的消息则不能在互联网这个大海上掀起什么波澜。所以，在构建双层社交网络时，要考虑到线下、线上社交网络各自的特点来进行网络的建立。

在当今社会，新媒体行业例如自媒体蓬勃发展，导致许多新闻在第一时间会被多个不同的传媒平台、同一传媒平台的多个不同媒体机构报道，而每个报道都有一定概率对一个用户造成影响。基于这样的特点，我们不难看出独立级联模型相较而言能更好地模拟现实信息传播情况。

1.2 国内外研究现状

Domingos[2]、Richardson[3] 首次提出了影响力最大化问题，旨在利用数据挖掘技术来寻找最佳的病毒式营销方案，初步确定了衡量影响力最大化的客观指标，例如聚类系数等。

Kempe[1] 首次将影响力最大化问题定义为离散优化问题，并且提出了运用独立级联模型和线性阈值模型的贪心算法（此算法得到的优化近似解近似比可达到$(1-1/e)$。由于贪心算法时间复杂度太高，不适用于大型网络求解。

Lescovec[4] 利用独立级联模型的次模性，在传统贪心算法基础上改进提出 CELF（Cost Effective Lazy Forward Algorithm）算法，最显著的特征是省略了大量节点的重复计算。这样做不仅可以大大提速（提高了将近 700 倍计算速度），同时具备通用性强的特点；缺点是算法时间复杂度依然太高，不适用于大型网络求解。

Kumar[5] 将复杂的影响模型转化为代理模型，提出了 CSR（Communities based Spreader Ranking）算法，算法中定义了社区模块化、社区多样性和社区密度三个度量指标，并利用这三个指标来对节点进行排序，以此来确定种子节点；Zhang[6] 注意到现实网络中存在的“富人俱乐部现象”——即高度节点强烈倾向于连接另一

个高度节点。为了解决这个问题，他们提出了基于投票机制提出了 VoteRank 算法来削弱邻居节点的连接能力。这些都是比较实用高效的尝试，美中不足是没有足够的理论支撑算法结果的准确性。

以上所提到的都是对单层网络的社交网络影响力最大化问题的研究，随着线上社交技术的发展与成熟，如今的社交网络已经逐渐发展成线上、线下社交并行的模式，上述研究已经无法解决问题。因此，针对双层网络甚至多层网络的研究地位越来越高。

胡斌[7] 等人从双层网络来进行 RBF（Radial Basis Function）神经网络的建模，同时采用粒子群优化算法（Particle Swarm optimization,PSO）进行调整和优化，得出以下结论：双层网络比单层网络更适合模拟信息传播的模型。此外，许多研究人员基于传染病动力学以及传染病模型（SIR），建立了许多信息传播模型。

Chen[8] 等人于 2010 年以独立级联模型为传播模型，提出了最大影响力树（Maximum Influence Arborescence,MIA）算法及其延申算法——排除最大影响前缀树（prefix excluding MIA,PMIA）算法。在算法里，通过节点的影响力最大化传播路径来对节点的影响力进行估计，再以此来选择种子集，该算法相较于传统的贪心算法，有着较低的时间复杂度，但是由于要对每个节点都建立树来计算，该算法的空间复杂度较高。总体来说，该算法为基于独立级联模型的影响力最大化问题提供了新的思路。

2014 年，Gao[9] 等人依据节点影响力的产生与其所处局部结构间的内在联系，创新性地开发出一种基于局部结构中心性的计算方法。该方法深度结合了节点自身及其相邻节点的拓扑属性与中心性特征，旨在对节点的影响力实现更为精确的量化评估。该方法被后续许多研究借鉴。

在研究社交网络影响力最大化问题中，对网络的边和节点的影响力进行评估与度量是十分重要的一环。早在 1950 年，为了刻画出节点的中心性（Centrality），A. Bavelas[10] 提出了接近中心性（Closeness Centrality）作为初步尝试，引发了人们对节点影响力建立度量准则的风潮；1979 年，Freeman[11] 提出了度中心性（Degree Centrality）、介数中心性（Betweenness Centrality）的概念，为量化节点重要性做出了巨大突破；Marchiori 和 Latora [12] 也在 2000 年提出了调和中心性（Harmonic Centrality），此外还有学者提出紧密中心性、网页排名中心性[13] 等等概念。这些量化指标丰富了节点影响力的评估度量准则，大大推进了影响力最大化问题的发展。

1.3 研究内容

本文采用独立级联模型为信息传播模型，根据现实里线下、线上的社交网络的特征建立合适的双层社交网络模型，探讨在这样条件下的影响力最大化问题。

1.4 论文结构

第 1 章为绪论，主要介绍了研究背景、问题研究意义、国内外研究现状等。

第 2 章为预备知识，主要介绍了社交网络、影响力最大化问题所涉及到的一系列概念的定义以及相关理论。

第 3 章为实验设计，主要阐述了强关系社交网络和弱关系社交网络的生成机制，对双层社交网络进行构建、对问题提出算法等等。

第 4 章为程序实现及结果分析，主要进行对数据结果进行整理与分析。

第 5 章为总结与展望，归纳了文章所做的工作与创新点，指出了文章的不足，并指明了进一步的研究方向。

2 理论知识

2.1 社交网络

定义 2.1.1：社交网络指一个图 G（V,E,P），其中 V 为节点集，E 为边集（可以为有向边集，也可以为无向边集），P 为概率集；每个用户都是一个不同的节点 v，用户之间的关系就是边 e，每条边都有一个概率 p，信息会沿着边按照对应的概率在节点之间传播。

本文考虑无向的社交网络的情况。

对影响力的研究既涵盖定性的解析维度，也包括定量的评估手段。虽然学术界至今为止尚未统一对影响力进行量化的方式，但是每种量化方法都有自己的理论保障。

要实现对节点影响力的定量衡量，关键在于设计一套理论可行且具有可操作性的方法。目前已有的基于网络拓扑结构的度量，大致可以分为“基于局部属性的度量”、“基于全局属性的度量”、“基于随机游走的度量”、“基于社团关系的度量”四类。下面将介绍“基于局部属性的度量”、“基于全局属性的度量”的部分概念。

2.2 基于局部属性对网络拓扑结构的度量

2.2.1 度、平均度

定义 2.2.1：节点的度（Degree）是用来刻画节点重要程度的统计量。在无向图中，节点 i 的邻边数目 k_i 称为该节点的度。

一个节点的度越大，代表其与更多的节点有关联，对信息传播的影响也相对较大，可称为该节点越重要；相反，如果一个节点的度越小，则说明该网络中与此节点相互联系的节点数越少。

定义 2.2.2：对一个网络中所有节点的度求平均值 $< k >$, 则 $< k >$ 就是平均度。假设无向图网络 G 中含有 N 个节点，则平均度 $< k >$ 为：

$$< k >= \frac{1}{N} \sum_{i=1}^{N} k_i$$

平均度的大小反映着网络的连接紧密程度，当平均度 $< k >$ 越大，则表示网络中每个节点的连边相对越多，网络连接紧密程度越高。

2.2.2 聚类系数、平均聚类系数与度中心性

定义 2.2.3：聚类系数指在网络中，与同一个节点连接的两个节点之间也进行相互连接的概率，是用来刻画网络局域性质的指标。在无向图中，可知节点 i 的度 k_i 即代表节点 i 与 k_i 个节点直接连接，且这 k_i 个节点之间可能存在的连接的边的个数最大为：$\frac{k_i(k_i-1)}{2}$，而实际存在的连接的边数为 M_i，则节点 i 的聚类系数表示为：

$$C_i = \frac{2M_i}{k_i(k_i-1)}$$

定义 2.2.4：平均聚类系数即为聚类系数的平均值：

$$C = \frac{1}{N}\sum_{i=1}^{N} C_i$$

平均聚类系数同样反映了网络中节点连接的紧密程度，当 $C=0$ 时，代表所有节点都是孤立点；而当 $C=1$ 时，表示网络中所有节点之间都进行连接。

定义 2.2.5：节点度中心性是节点的度和该节点若与图内所有其他节点建立连接的总边数的比值：

$$DC_i = \frac{k_i}{N-1}$$

度中心性是刻画节点中心性的最直接度量指标，与节点的度大小直接有关联，即一个节点的度越大，则该节点度中心性越大。中心性越大，则节点越重要。

2.3 基于全局属性对网络拓扑结构的度量

2.3.1 平均距离、节点介数与介数中心性

定义 2.3.1：设 d_{ij} 为有 N 个节点的网络 G 中任意两个节点 i 和 j 之间的距离为任意两个节点 i 和节点 j 之间的最短路径长度。

如果两个节点 i, j 之间没有通路（即在图 G 中从节点 i 出发无法到达节点 j），则定义它们之间的距离 $d_{ij}=\infty$。

距离反映着节点之间的信息传播的快慢程度。如果距离越长，则信息传播的速度越慢。

定义 2.3.2：网络 G 的平均距离 L 为所有节点对之间的距离的平均值, 则：

$$L = \frac{1}{N^2}\sum_{j=1}^{N}\sum_{i=1}^{N} d_{ij}。$$

平均路径长度反映了整个网络的节点离散程度，在不同的问题背景下，平均路径长度往往含有不同的内涵，例如在研究信息传播的问题中，平均路径长度往往反映着网络整体信息传播速度的快慢，而在城市规划等问题中，平均路径长度代表着不同站点之间的平均距离。

定义 2.3.3：节点介数（Betweenness）指网络里任意两个节点之间的最短路径通过该节点的条数占最短路径总数的比例。

节点介数通过度量节点作为“桥梁”的作用，反映相应节点在整个网络中的影响力。

定义 2.3.4：节点的介数中心性以网络中每队顶点的最短路径数做分母，其中经过该节点的最短路径数做分子，进行求和。设 n_{ij} 为 i 节点到 j 节点的最短路径数，$n_{ij}(m)$ 代表从 i 节点到 j 节点经过 m 节点的最短路径数。则节点的介数中心性为：

$$BC_k = \sum_{i \neq j \neq m, i<j} \frac{n_{ij}(m)}{n_{ij}}$$

介数中心性的概念最早由 Freeman[11] 在 1977 年提出，并在 1979 年被定义，它刻画的是网络中信息传播过程里经过该节点的频率，该指标越大，则该节点在信息传播中越“繁忙”，该节点也就越重要。

2.4 影响力最大化问题

定义 2.4.1：影响力最大化即在社交网络中求得一组有影响力的用户节点作为初始活跃节点（种子节点），通过这些种子节点向社交网络中传递信息，使得信息的传播范围可以最大化（即最终被影响的节点个数最多）。主要分为两种类型：

（1）给定节点数 k，选择出 k 个节点作为种子集使得种子集能影响的节点数最多。

（2）给定所要求产生的影响力，找到满足条件的最小节点集合。

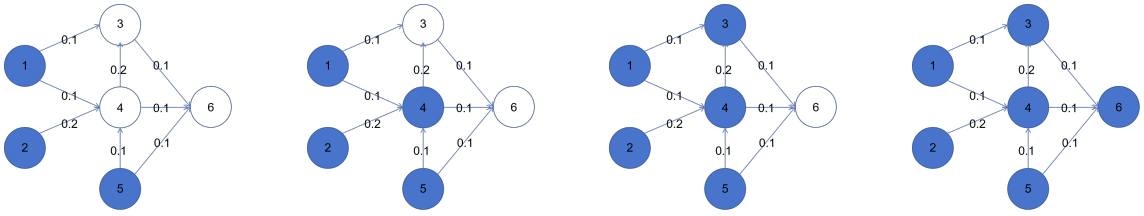
本文主要研究第一种影响力最大化问题。

2.5 传播模型

2.5.1 线性阈值（Linear Threshold Model, LT）模型

线性阈值模型给每个节点赋予了一个阈值，该阈值反映对应节点的受影响的难易程度。对于处于未激活态的节点 i, 它有自己的阈值 θ_i, 与节点 i 相邻的每个节

点 j 以非负的权重 w_{ji} 对节点 i 产生影响，并且对于 i 节点的全部邻居节点 j, 有 $\sum_j w_{ji} \leq 1$。此时，当 $\sum_j w_{ji} \geq \theta_i$ 时，节点 i 激活被，即网络中个体的状态依赖于其所有邻居节点的共同作用。且节点 i 的活跃邻居节点可以多次参与激活 i。

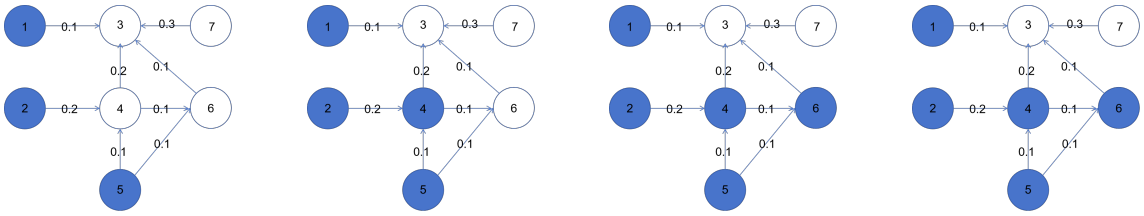


(a) 第一步　(b) 第二步　(c) 第三步　(d) 第四步

图 2.1　阈值 $\theta = 0.25$，种子集为 {1,2,5} 的线性阈值模型示意图

2.5.2　独立级联（Independent Cascade, IC）模型

独立级联模型是一个概率模型，其中每一个节点有两种状态：活跃状态与非活跃状态，处于非活跃状态的节点有概率被处于活跃状态的邻居节点影响激活，从而进入活跃状态；已经进入活跃状态的节点无法恢复非活跃状态。在传播最开始时，只有种子集里的节点处于活跃状态；在 t 时刻为活跃状态的 i 节点将在 $t+1$ 时刻试图激活邻居节点 j，激活成功的概率为 $p_{i,j}$；i 节点对它的所有邻居节点都有且只有一次激活机会，无论激活成功与否，i 节点都丧失了再次激活同一节点的能力。这一类节点被称为无影响力的活跃节点。



(a) 第一步　(b) 第二步　(c) 第三步　(d) 第四步

图 2.2　种子集为 1,2,5 的独立级联模型示意图

2.5.3　对比

两种传播模型都在一定程度上体现了信息扩散的特点，并且都引入了随机性，但是有着明显差别。

线性阈值模型的特点有：第一，线性阈值模型以非活跃的节点为中心 (receiver-centered)，通过观察一个节点的所有邻居节点来判断这个节点是否被激活；第二，在信息传播过程种网络中一个节点激活与否由节点的所有邻居节点共同作用而决定；

第三，当模型中每个节点的阈值确定下来后，整个传播过程也确定下来，不再存在随机性。

独立级联模型的特点有:第一,独立级联模型以活跃节点为中心(sender-centered),当一个节点处于活跃状态后将会以一定概率去激活每一个邻居节点；第二，在信息传递过程中，每一个处于激活态的节点独立地尝试激活每一个邻居节点；第三，采用独立级联模型后整个信息传播过程是随机的。

线性阈值模型考虑到了节点共同影响的联合效应，即可能一个邻居无法激活节点，但是在多个邻居共同作用下可以激活该节点，这很好的反映了人的从众行为。但是，线性阈值模型的随机性完全由各个节点的阈值确定，当每个节点的阈值确定的时候，该模型也就失去了随机性。

由此可见，独立级联模型更符合多种社交媒体蓬勃发展、信息爆炸的现状。许多事件，例如在线消息的传播和病毒的传播等，都有着独立传播的特性。因此本文研究采用独立级联模型来进行研究。

2.6 贪心算法

2.6.1 单调性与次模性

定义 2.6.1 (**单调性**)：如果集合函数 f：$2^V \rightarrow R$ 满足对任何子集 $S \subset T \subset V, f(S) \leq f(T)$, 则称其为单调非减的；集合函数 f：$2^V \rightarrow R$ 满足对任何子集 $S \subset T \subset V, f(S) \geq f(T)$, 则称其为单调非增的。如果集合函数 f 为单调非增的或单调非减的，则可称其为单调的。

定义 2.6.2 (**次模性**)：如果集合函数 f：$2^V \rightarrow R$ 满足对任何子集 $S \subset T \subset V$ 和任何的 $v \in V \backslash T$ 都有 $f(S \cup v) - f(S) \geq f(T \cup v) - f(T)$, 则称 f 满足次模性

由于贪心算法的特性，当传播模型在满足单调性和次模性时，采用贪心算法可以得到近似比逼近（$1 - 1/e - \varepsilon$）的优化近似解（Kempe[1] 等人文章已经做出了证明)。而线性阈值模型与独立级联模型可以很好满足这两个性质。虽然有多种其他算法来解决影响力最大化问题，但是贪心算法依然是准确率最高的，于是本文将建立在上述两种的传播机制模型的基础上采用合适的贪心算法进行研究。

2.6.2 贪心算法可行性证明

定理 2.6.1：[14] 对于满足次模性，单调性，且在定义域上值域是非负的函数 f，使用贪心算法可以获得近似比至少为 $(1 - 1/e - \varepsilon) \approx 63\%$ 的优化近似解。

原始问题为：如果存在一个有限集合 X，$S \subset X$，存在一个具有次模性和单调性的函数 f,k 是一个介于 1 和集合 X 大小之间的整数，寻找一个 k 大小的集合 S 使得 $f(S)$ 尽可能大。

引理 2.6.1：假设 $S_1, S_2, ..., S_l$ 是贪心算法每一轮迭代得到的集合，则当 $l > k$ 时，可以得到：

$$f(S_l) \geq (1 - e^{-\frac{l}{k}}) \max_{T:|T|\leq k} f(T)$$

证明：设 S_i 是利用贪心算法在第 i 轮迭代后得到的集合，S^* 是大小为 k 的情况下的实际上的最优集合，即 $S^* = \{x_1^*, x_2^*, ..., x_k^*\}$。

对于 $i < l$，通过 f 的次模性，我们可以知道

$$f(S^*) \leq f(S^* \cup S_i)$$

接下来我们可以将其分解成添加了单个元素的边际效应的总和。

$$f(S^* \cup S_i) = f(S^* \cup S_i) + f(\{x_1^*, ..., x_{k-1}^*\} \cup S_i) - f(\{x_1^*, ..., x_{k-1}^*\} \cup S_i)$$

删除 x_{k-1}^*, x_{k-2}^*，直至将 S^* 中所有元素全部删除殆尽，则可以得到：

$$f(S^* \cup S_i) = f(S_i) + \sum_{j=1}^{k} (f(\{x_1^*, ..., x_j^*\} \cup S_i) - f(\{x_1^*, ..., x_{j-1}^*\} \cup S_i))$$

则思路就从“向已经存在部分 S^* 元素的 S_i 中继续加入 S^* 元素”转化成“向 S_i 集合中继续加入一些 S^* 元素”。则可以上述等式可以变为：

$$f(S_i) + \sum_{x \in S^*} f(S_i \cup x) - f(S_i)$$

由于采用贪心算法，S_{i+1} 是迭代出来的，则可以得到：

$$f(S_i) + \sum_{x \in S^*} f(S_{i+1}) - f(S_i) = f(S_i) + k(f(S_{i+1}) - f(S_i))$$
$$f(S^*) - f(S_i) \leq k(f(S_{i+1}) - f(S_i))$$

假设 $a_i = f(S^*) - f(S_i)$，则有：

$$a_{i+1} \leq (1 - \frac{1}{k}) a_i$$
$$a_l \leq (1 - \frac{1}{k})^l a_0$$

又因为：

$$a_0 \leq f(S^*)$$
$$1 - x \leq e^{-x}$$

可以得到：

$$a_l = f(S^*) - f(S_l) \leq e^{-\frac{l}{k}} f(S^*)$$
$$f(S_l) \geq (1 - e^{-\frac{l}{k}}) f(S^*)$$

引理 2.3.1 得证。特殊的，当 $l = k$ 时，定理 2.3.1 显然成立。至此，影响力最大化问题的贪心算法的可行性得到了保障。

□

2.6.3 贪心算法伪代码

贪心算法基于影响力最大化问题的单调性（Monotonicity）和次模性（Submodularity）特点，使用爬山策略（Hill Climbing），其基础算法框架如下：

算法 1 贪心算法（k,f）：普通贪心算法

Input: k: 返回子集的大小；f: 满足单调性和次模性的集函数

Output: 选定子集

初始化 $S \leftarrow \varnothing$;

for $i = 1, 2, \ldots, k$ **do**

  $u \leftarrow \arg\max_{w \in V \setminus S} f(S \cup \{w\}) - f(S)$

  $S \leftarrow S \cup \{u\}$

end

return S

事实上，我们无法得知节点集 S 的真实传播范围，往往采用蒙特卡洛方法估计它的影响范围，则真实情况下算法框架如下：

算法 2 MC-贪心算法（k,f）：蒙特卡洛贪心算法

Input: G: 带有 IC 或 LT 模型的弧权值的社交图谱； k: 种子集大小

Output: 选定种子集

初始化 $S \leftarrow \varnothing$;

for $i = 1, 2, \ldots, k$ **do**

  蒙特卡洛估计 $\arg\max_{w \in V \setminus S} (S \cup \{w\}, G) \rightarrow u$

  $S \leftarrow S \cup \{u\}$

end

return S

Function *蒙特卡洛估计 (S,G)*:

  $count \leftarrow 0$

  for $j = 1, 2, \ldots, R$ **do**

    模拟在种子集取 S 情况下图 G 上的扩散情况

    模拟扩散后得到的处于激活态的节点 $\rightarrow n_a$

    $count \leftarrow count + n_a$

  end

  return $count/R$

end

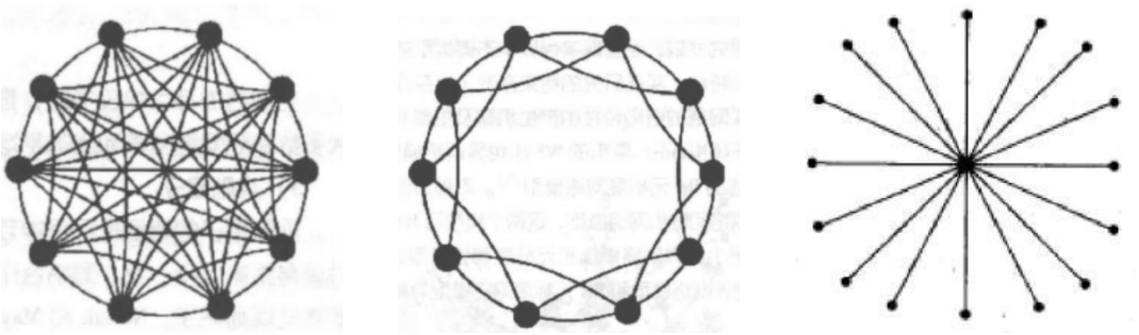
贪心算法的算法思想简单，具有易于理解与重复操作、求解稳定性强、准确率高的优点，优化近似解可以达到（$1 - 1/e - \varepsilon$）的近似比；但是这一切好的结果的前提是传播机制需要满足次模性与单调性，仅仅适用于特殊的模型，具有一定局限性，并且算法时间复杂度高，效率相对较低。

2.7 经典网络模型

2.7.1 规则网络

规则网络是结构最为简单的早期复杂网络。通常来说，规则网络中的任意两个节点之间的连接遵循某种既定的规则，且每个节点的近邻数目通常相同。

常见的规则网络包括括全局耦合网络、最近邻耦合网络和星型耦合网络等。



(a) 全局耦合网络　　(b) 最邻近耦合网络　　(c) 星型耦合网络

图 2.3　规则网络图

由于大多数规则网络具有较大的平均距离长度和聚类系数, 因此无法反映现实中结构的异质性及动态增长性，于是本文不采用规则网络。

2.7.2　随机网络

Paul Erdős 和 Alfréd Rényi[15] 提出了随机网络，模型构建的主要想法为：以固定或随机的概率 p 将网络中的 N 个节点进行连接。算法伪代码如下所示：

算法 3 随机网络构建算法

Input: G: 包含网络所有节点的散点图;p: 节点间连接的概率

Output: 一个 ER 随机网络

while *存在节点未进行判定* **do**

　在网络中随机选取一对没有任何连边的节点;

　生成随机数 r;

　if $r < p$ **then**

　　该对节点间建立连接;

　else

　　该对节点间不进行连接;

　end

end

return G

在随机网络中，平均度 $< k >$ 为：

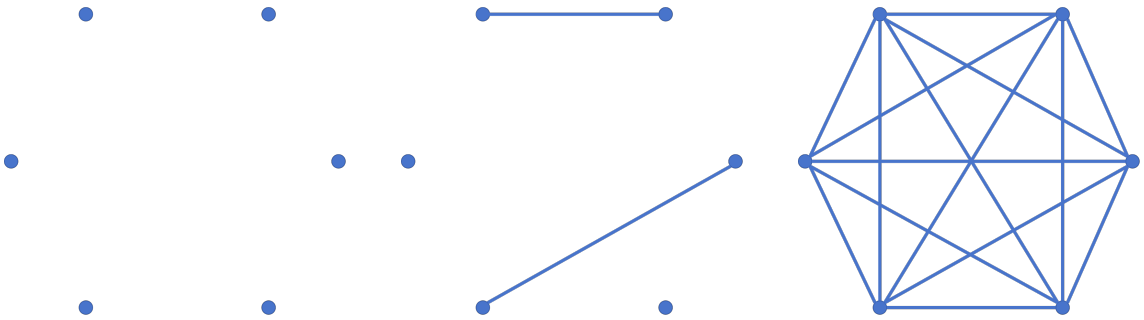
$$< k >= p(N-1) \approx pN$$

由随即网络的构建算法可知，各个节点之间连接的概率是相等的，则观察节点 i，其与其他节点连接的概率也是 p，可以得出随机网络的聚类系数为：

$$C \approx p = \frac{< k >}{N}。$$

由此可见，随机网络的聚类系数与节点总数 N 有关；而根据日常经验可以分析，真实的网络的聚类系数应该与节点总数 N 无关，与邻居节点数有关。因此，随机网络也并不能很好的模拟真实网络。

下图展示节点数 $N = 6$ 时，不同 p 下的随机网络生成情况。



(a) $p = 0$　　(b) $p = 0.5$　　(c) $p = 1$

图 2.4　N=6 时，不同 p 值的随机网络图

2.7.3　小世界网络

真实网络具有一定的规律，又具有一定的随机性，所以网络结构应该处于规则网络与完全随机网络之间。

为了更好模拟真实网络，1998 年，D.J.Watts 和 S.H.Strogatz[16] 基于小世界现象提出了小世界模型，并被视为从规则网络到完全随机网络的过渡。小世界现象是指，地球上的任意两个人之间都可以通过少数几个中间人（平均为 6 个）来建立联系。

由于小世界网络发展众多，本文主要介绍最基础的小世界网络——WS 小世界网络。该模型最大的特点为：“随机化重连”，具体算法如下：

算法 4 小世界网络构建算法

Input: G: 包含网络所有节点（数量为 N）的散点图;p: 节点间连接的概率

Output: 一个小世界网络

以 N 个节点建立一个的最近邻耦合网络，节点相连构成一个环形，其中每个节点都与左右相邻的各 $\frac{K}{2}$ 个节点连接，其中 $N \gg K \gg \ln N \gg 1$,K 为偶数

while *存在边未进行判定* **do**

    选取为判定的边（称为 AB）的一个节点 A;

    生成随机数 r;

    if $r < p$ **then**

        将 B 换做网络里随机一个与顶点（除 A 外的点，且该顶点本身未与 A 连接）连接;

    else

        不进行连接;

    end

end

return G

通过该算法可以生成 $\frac{pNK}{2}$ 条边使得每个节点都可以与距离较远的节点相关联。

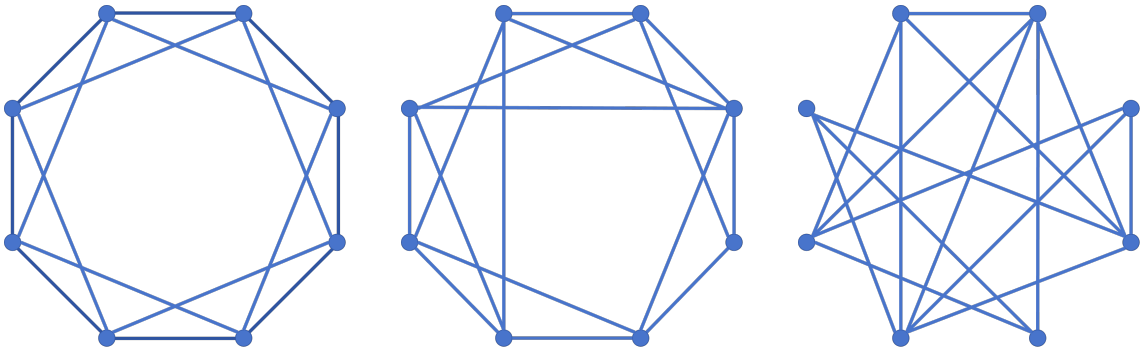
由小世界网络的构建算法可知:

当 $p = 0$ 时，该网络即为完全规则网络;

当 $p = 1$ 时，该网络为完全随机网络;

通过调节 p 的值可以实现从完全规则网络到完全随即网络的过渡。

下图展示节点数 $N = 8$ 时，不同 p 下的小世界网络生成情况。



(a) $p = 0$　　(b) $p = 0.5$　　(c) $p = 1$

图 2.5　$N = 8$ 时，不同 p 值的小世界网络图

此时，得到的小世界网络的聚类系数为:

$$C = \frac{3(k-2)}{4(k-1)}(i-p)^3。$$

2.7.4 无标度网络

Barabasi 和 Albert[17] 在 1999 年指出前人提出的网络模型，无论是随机网络还是小世界网络都未考虑到真实网络的增长性。真实网络的节点数往往会随着网络的扩张而不断增长。同时，先前的网络也未曾考虑“马太效应”，即在真实的网络中，新加入的节点往往会优先加入连接数较多的节点。

为了满足这两个真实网络的特性，Barabasi 等人[17] 证明了无标度网络更符合真实网络。该网络的构建算法如下：

算法 5 无标度网络构建算法

Input: G: 包含网络 N 个节点（数量为 N）的网络图

Output: 一个无标度网络

while *要加入新的节点* **do**

  将新节点与原先存在的 N 个节点中的 M 个节点连接（$M \leq N$）

  while M **do**

    在判定旧节点 i 与新节点连接时，取 $p = \dfrac{k_i}{\sum_j k_j}$，其中 k_i 为节点 i 的度；

    生成随机数 r；

    if $r<p$ **then**

      新节点与旧节点 i 进行连接

    else

      不进行连接;

    end

  end

end

return G

在经过 t 轮循环后，该算法将产生一个具有 $M+t$ 个节点，有 Nt 条边的网络。

2.8 熵权法

2.8.1 信息熵

信息熵是 1948 年 Shannon[18] 从热力学中获得灵感，用于衡量信息量的概念，是衡量一个系统有序程度的度量。如果一个系统越混乱，则它的信息熵越高；一个系统越有序，则它的信息熵越低。

定义 2.8.1：信息中排除了冗余后的平均信息量称为“信息熵”。

假设离散随机变量 X 的概率分布为 $P\{X=x_i\}=p_i(i=1,2,...,k)$，满足 $p_i \geq 0$，

$\sum_{i=1}^{k} p_i = 1$。

则对应的信息熵的定义为：

$$H\ （X）=-\sum_{i=1}^{k} p_i \log_2 p_i。$$

2.8.2 熵权法

根据信息熵的定义，我们可以发现，当一个指标的信息熵越大，就代表着该指标信息的混乱程度越高，则它的不确定性就越大，且所包含的信息量越小，则该指标的综合评价能力就越弱，那么该指标的权重就应该越小。

定义 2.8.2：是一种根据各指标数值变化对整体的影响，计算指标的熵值，进而确定权重的赋权方法。

该方法可以消除人为主观赋值带来的偏差，规避主观因素的影响，提高赋权的客观性和准确性。

熵权法的步骤如下：

第一步：数据标准化。假设样本由 m 个样本与 n 个指标构成：

$$X = \begin{bmatrix} x_{11} & \cdots & x_{1n} \\ & \ddots & \vdots \\ x_{m1} & & x_{mn} \end{bmatrix}_{m*n}$$

则有正向指标标准化：

$$X_{ij}^{1} = \frac{X_{ij} - \min(X_{1j}, ..., X_{mj})}{\max(X_{1j}, ..., X_{mj}) - \min(X_{1j}, ..., X_{mj})}。$$

有负向指标标准化：

$$X_{ij}^{1} = \frac{\max(X_{1j}, ..., X_{mj}) - X_{ij}}{\max(X_{1j}, ..., X_{mj}) - \min(X_{1j}, ..., X_{mj})}。$$

则经过数据标准化，可以得到新的数据：

$$X^{1} = \begin{bmatrix} x_{11}^{1} & \cdots & x_{1n}^{1} \\ & \ddots & \vdots \\ x_{m1}^{1} & & x_{mn}^{1} \end{bmatrix}_{m*n}$$

第二步：计算比重。计算第 j 个指标的第 i 个项目的比重 P_{ij}：

$$P_{ij} = \frac{x_{ij}^{1}}{\sum_{i=1}^{m} x_{ij}^{1}} (i = 1, 2, ..., m; j = 1, 2, ..., n)。$$

第三步：计算熵值。计算第 j 个指标的熵值 e_j:

$$e_j = -\frac{\sum_{i=1}^{m} P_{ij} * \ln P_{ij}}{\ln m}$$

第四步：计算变异系数。计算第 j 个指标的变异系数 d_j:

$$d_j = 1 - e_j (j = 1, 2, ..., n)$$

第五步：计算熵权。计算第 j 个指标的权重 w_j 为：

$$w_j = \frac{d_j}{\sum_{j=1}^{n} d_j}$$

2.9 本章小结

本章主要介绍了社交网络相关的基础概念，分析了两种传播模型，并进行特点总结与对比；介绍了解决影响力最大化问题的贪心算法；根据时间顺序，将各种网络模型的基础特征和算法进行介绍，为后文的实践打下坚实的理论基础。

3 实验设计

3.1 单层社交网络构建

3.1.1 线下社交网络——强关系社交网络

本文认为，线下的社交关系往往是两个个体通过密切的交流与合作而产生的关系，在或多或少的方面有更为深入密切的交集，比如家庭关系、朋友关系等。本文假设每个个体有六个社会经济属性（性别、年龄、受教育程度、民族、地理位置、户口），对于个体 i，这六个社会经济属性分别记为 $C_i^1, C_i^2, C_i^3, C_i^4, C_i^5, C_i^6$。每个社会经济属性的比重通过熵权法进行确立，定义为 $\lambda_k (k = 1, 2 \ldots\ldots 6)$。

由此，我们可以定义两个个体 i、j 之间的“差异度” R_{ij}：

$$R_{ij} = \sqrt{\sum_{k=1}^{6} \lambda_k (C_i^k - C_j^k)^2}$$

若个体 i 与个体 j 之间的“差异度” R_{ij} 低于某个值（本文取 0.3），则可以认为二者之间的关系是强关系，可以在社交网络中将节点 i 与节点 j 之间建立连接。由此，可以建立出线下社交网络——强关系社交网络，定义为 G_1。

3.1.2 线上社交网络——弱关系社交网络

相较于线下的社交网络，线上的社交网络往往交往的不那么深入，或是只有在某一方面有共同交集，例如有相同爱好等。目前为止，互联网上的社交网络往往都是属于弱差异度社交网络，本文定义为弱关系社交网络。目前，已有部分文献证明大部分的线上社交网络的拓扑结构具有扁平化和无标度的特性[19]，于是本文将采取无标度网络来模拟生成弱关系社交网络 G_2。

3.2 双层社交网络构建

双层社交网络 G 是将线下社交网络与线上社交网络结合起来的社交网络。为了简化模型，我们默认一个线下社交网络的个体只对应一个线上社交网络的个体，即一个人/用户有且仅有一个线上社交账号。下图展示一个双层社交网络，在图形中，每个圆圈代表一个节点，每条实线代表两个节点之间存在关系，虚线代表双层社交网络之间的对应关系，即 G_1 的人对应在 G_2 中的社交媒体上的账号。

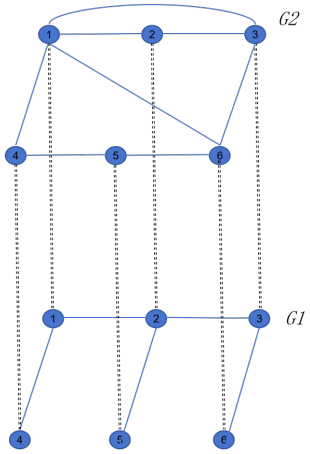


图 3.1　双层社交网络 G 示意图

由图 3.1 可知，在双层社交网络 G 的子网络——强关系社交网络 G_1 中节点 1 与节点 6 并未建立联系，在子网络——弱关系社交网络中建立了联系。这是由于 G_1 与 G_2 各自的拓扑特征和构建算法决定的。同时这也是符合实际的——在现实生活中没有交集的两个人，可以通过在网络上聊天等网络社交活动而建立起联系。

本文建立在单层社交网络构建的基础上，提出了构建双层社交网络的算法，具体算法如下：

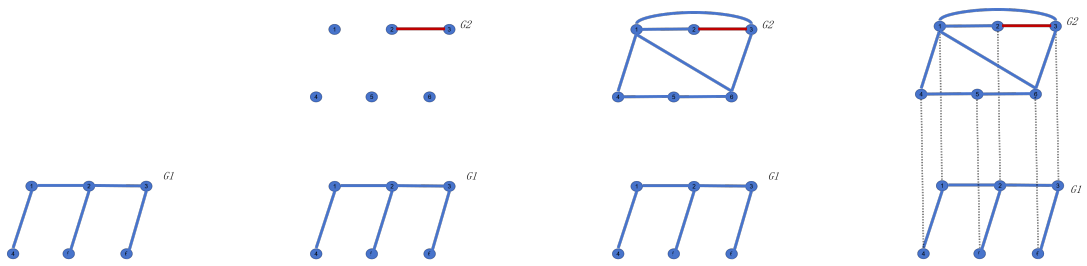
第一步：确定要分析的 N 个个体，根据官方发布的人口普查数据赋予每个个体社会经济属性 $C^i(i = 1, 2, ..., 6)$，并计算差异度 R_{ij}. 当差异度小于给定的值时，在个体 i 和个体 j 之间产生连边，由此建立起线下社交网络——“强关系”社交网络 G_1。

第二步：将线下社交网络 G_1 中的全部节点复制到 G_2 中，以一定概率将 G_1 中的连边映射到 G_2 中，形成 G_2 的初始网络。

第三步：采用无标度网络的构建方法，将 G_2 的初始网络中已经存在连边的节点视作旧节点，将未连边的节点视作新节点，以旧节点的度占网络里所有度的比例为概率进行连边。

第四步：将两层网络间对应的节点连接，形成双层社交网络。

双层社交网络构建的示意图如下所示：



(a) 第一步 (b) 第二步 (c) 第三步 (d) 第四步

图 3.2 双层网络构建过程示意图

这样形成的双层社交网络，信息将同时沿着 G_1 和 G_2 传播。

伪代码如下所示：

算法 6 双层社交网络构建算法

Input: G: 包含网络所有节点（数量为 N）的散点图；
t: 强关系社交网络连接的判定值；
p_1：在子图 G_1 生成 G_2 时映射边的概率；
Output: 一个双层社交网络
通过统计数据确定每个个体 i 的社会经济属性 $C_i^k(k = 1, 2..., 6)$；
根据 C_i^k 计算每个个体 i，j 之间的“差异度” R_{ij}；
while *存在两个节点未进行判定* **do**
  if R_{ij}<t **then**
    节点 i 与节点 j 之间建立连接；
  else
    不进行连接；
  end
end
将产生的子图 G_1 的全部个体复制到 G_2 中；
以概率 p_1 将子图 G_1 中的边映射到子图 G_2 中；
将子图 G_2 中已存在连边的节点视为旧节点，未存在连边的节点视为新节点；
利用无标度网络的构建方法，构建完整子图 G_2；
将子图 G_1 与子图 G_2 中对应的节点连接，构建出双层社交网络 G；
return G

3.3 贪心算法改进设计

目前已经建立起了双层社交网络模型，那么如何在双层社交网络上进行影响力最大化问题的求解呢？本文将单层社交网络的贪心算法进行改进，使其可以更好地运用在双层社交网络上。

在现实情况里，信息的传播往往是线上与线下同时进行的，且任何一个个体，无论是在线上社交媒体上被影响还是在线下社交活动中被影响，这个个体都获取了传播的信息。于是，我们可以得出，最终的影响力最大化范围是线上社交网络的影响力最大化范围和线下社交网络影响力最大化范围的并集。

基于这样的想法，我们将贪心算法进行改进，具体操作为，将原本每一轮的遍历单层网络来判断种子集的影响力范围部分，改为同时在 G_1 和 G_2 上遍历每一个节点，并令 G 每一轮的影响范围为 G_1 与 G_2 的影响范围的并集（G_1 与 G_2 的对应节点视为同一个节点）。当有两个节点 i、j 在统一轮迭代中在 G 上的影响范围是一样的时，考虑到 G_1 作为线下强关系社交网络，节点之间联系更为紧密，因此取在 G_1 中的影响范围较大的那个节点加入种子集；若两个节点在 G 和 G_1 上的影响范围都一样大，则取其中任意一个节点加入种子集。

改进的贪心算法的伪代码如下所示：

算法 7 贪心算法（k,f_1,f_2）：双层网络贪心算法

Input: k: 返回子集的大小;
f_1: 满足单调性和次模性的集函数
f_2: 满足单调性和次模性的集函数
Output: 选定子集
初始化 $S \leftarrow \varnothing$;
for $i = 1, 2, \ldots, k$ **do**
  $u \leftarrow \arg\max_{w \in V \setminus S} (f_1(S \cup \{w\}) - f_1(S)) \cup (f_2(S \cup \{w\}) - f_2(S))$
  $S \leftarrow S \cup \{u\}$
end
return S

3.4 实验参数选取

在线下社交网络 G_1 中，本文通过模拟数据，根据模拟结果确定个体 i, j 之间的差异度 R_{ij} 来判定 i, j 之间是否建立联系.

在将 G_1 的边映射到 G_2 时，取映射的概率 p_2=0.5。

在采用独立级联模型进行传播时，为了衡量每个节点的影响力大小，考虑到既要关注每个节点的本身特征，又要关注到每个节点的在整个网络中发挥的作用[20]，本文选择将“度中心性”和“介数中心性”两个指标分别归一化后以一定比例（可调整，默认分别乘以 0.6、0.4）混合，以此作为每个节点激活处于非激活态的邻居节点的概率（即影响力）。

3.5 本章小结

随着线上社交媒体的快速发展，人们交流信息的方式也变得越来越多，但线下社交网络上的交流依然占据主导地位，线下社交网络建立的关系也更加稳定牢固。此外，线上社交网络往往建立在线下社交网络的基础上。为了反映出这些特点，本章分别模拟构建了符合实际的线下社交网络和线上社交网络，并将其结合成双层社交网络，为后续的实验提供了坚实的理论基础。

4 程序实现及结果分析

4.1 统计数据获取

本文将每个个体的六个经济社会属性定位：性别（C_i^1），年龄（C_i^2），受教育程度（C_i^3），民族（C_i^4），地理位置（C_i^5），户口（C_i^6）。通过数据搜索，本文采取 2021 年全国第七次人口普查[21] 的数据来进行实验。具体数据如表 4.1 所示。

表 4.1 2021 年全国第七次人口普查数据 (%)

性别 C^1	男	女
占比	51.24	48.76

年龄 C^2	少年	中青年	老年
占比	17.95	63.35	18.70

受教育程度 C^3	文盲	初级	中级	高级
占比（少年）	2.70	96.90	0.40	0
占比（中青年）	5.50	73.75	15.85	4.90
占比（老年）	47.60	46.30	4.00	2.10

民族 C^4	汉族	少数民族
占比	91.11	8.89

地理位置 C^5	东	南	西	北	中	东南	东北	西南	西北
占比	22.86	11.04	0.23	11.77	20.46	2.74	8.42	20.95	1.53

户口 C^6	城市	农村
占比	63.89	36.11

4.2 相关参数计算

由上述表格内的统计数据，我们可以计算出实验所需的各个参数：

利用熵权法计算每个指标权重，则每个指标的权重组成的权重矩阵为：

$$\lambda = \begin{bmatrix} 0.0003 \\ 0.1285 \\ 0.2894 \\ 0.4270 \\ 0.1123 \\ 0.0425 \end{bmatrix} \tag{4.1}$$

同时，将每个指标进行赋值，从左往右，从 1 开始赋予整数，例如指标“性别 C_1”中男性赋予 1，女性赋予 2。在根据每个指标内概率给每个样本分配社会经济属性后，再对全体样本数据进行归一化处理，得到最终实验样本数据。

4.3 预实验

本文将先采取 20 个节点的情况进行分析作为预实验，如果实验结果较好，则采取总样本数为 500 进行实验分析。

根据人口普查数据概率随机生成 20 个样本，每个样本含有六个指标。归一化后得到一个 $20 * 6$ 的矩阵。（具体数据请见附录 A.1）

计算这 20 个样本的差异度，我们将得到一个 $20 * 20$ 的差异度矩阵 R，其中 R_{ij} 为样本 i, j 之间的差异度。将差异度数据统计下来，制作成差异度分布直方图（图中已将无意义的 $R_{ii} = 0$ 去除），见图 4.1。

有图像我们可以选取，当差异度 R_{ij} 小于等于 0.4 时，可以将个体 i, j 在 G_1 连接起来。

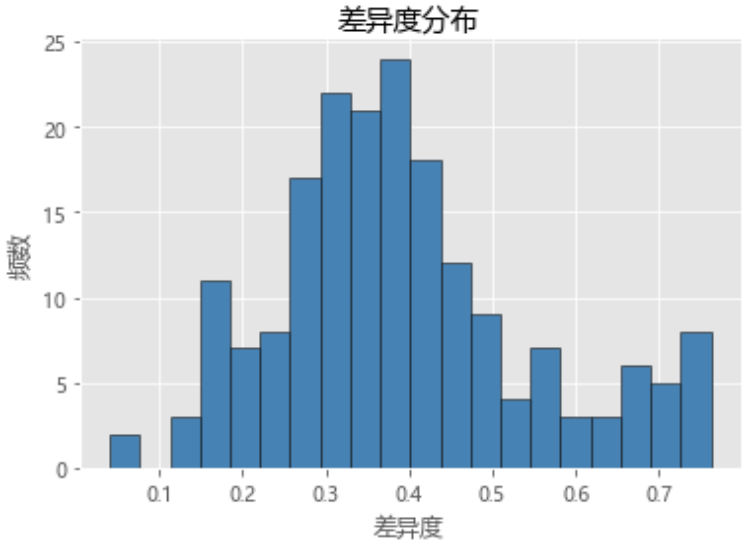


图 4.1 差异度分布直方图

此时，根据第三章中构建强关系社交网络的算法，我们可以构建出样本数为 20 时的强关系社交网络 G_1，再根据我们建立的构建双层社交网络的算法，构建弱关系社交网络 G_2 的初始网络，再通过无标度网络构建方法构建完整的弱关系社交网络 G_2。具体过程由图 4.2 展示。图像内节点的大小由该节点的度决定，即节点的度越大则该节点越大。

(a) 强关系社交网络网络 G_1

(b) 弱关系社交网络 G_2 初始网络

(c) 弱关系社交网络 G_2 最终网络

图 4.2　样本数为 20 的双层社交网络生成图

从图 4.2（a）中我们可以看见，左下角的节点 14 是孤立的节点，即它的度是 0；在图 4.2（b）中，左下角的节点 14 依然是孤立的节点；而从 G_2 最终网络图 4.2（c）可以看见，节点 14 有了一个连边，并且是与右下角的度比较大的节点 19 相连接。

由于独立级联传播机制是概率传播机制，在不同实验时同一个节点的激活状态不一定相同，因此贪心算法寻找种子集时具有不确定性。进行预实验时不限制寻找的种子集规模大小，旨在寻找到能影响整个网络的最小的种子集。对上面随机生成的 20 个样本进行多次实验，发现得到的种子集规模大小稳定在 3 附近。在这里将一定出现在种子集里的节点称为关键种子节点，则在上面生成的 20 个样本中关键种子节点为节点 1、节点 3，在种子集里出现过的非关键种子节点为节点 11、节点 13。在采用节点 1，节点 3，节点 11，节点 13 作为种子集时，将在 G_1 里进行 2 轮扩散，其中第一轮扩散后将共存在 18 个处于激活态的节点，在第二轮扩散后共存在 19 个处于激活态的节点；在 G_2 里进行 2 轮扩散，其中第一轮扩散后共存在 12 个处于激活态的节点，第二轮扩散后共存在 20 个处于激活态的节点。

4.4 实践结果展示与分析

根据人口普查数据概率随机生成 500 个样本，每个样本含有六个指标。归一化后得到一个 500*6 的矩阵。首先进行不限制种子集规模，要求使得 500 个节点都被激活的实验。由于独立级联传播模型是概率传播模型，每次实验的结果都不尽相同，于是本文进行了五次实验，力求得到相对稳定的结果。

根据实验结果，可以发现平均需要五个节点作为种子节点来使得 500 个样本都被激活；这些种子节点平均在 G_1 里进行 7 轮信息传播扩散，平均的最大传播范围是 496 个节点；在 G_2 里平均进行 7.6 轮信息传播扩散，平均最大传播范围是 478.8 个节点。

取其中一次实验数据进行展示:

种子集（节点序号）	1	16	63	115	97
G_1 中扩散轮次	9				
G_2 中扩散轮次	6				

G_1 中每一轮扩散范围（激活节点总数）	441	465	470	476	483	490	493	494	494

G_2 中每一轮扩散范围（激活节点总数）	336	463	469	475	478	478

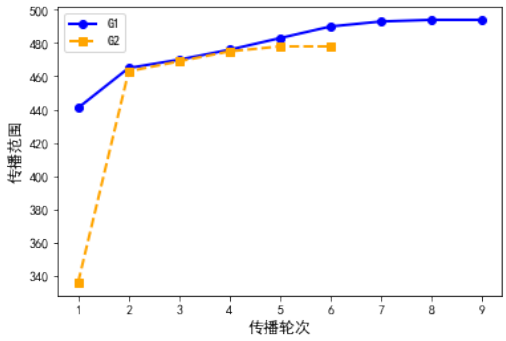


图 4.3 500 个随机样本（不控制种子集规模）信息扩散示意图

再次随机生成 500 个样本，此次实验控制种子集规模为 3，进行五次实验取平均值，可以得到在 G 中平均传播范围是 492.4 个节点；三个种子节点平均在 G_1 里进行 7.8 轮信息传播扩散，平均的最大传播范围是 473.4 个节点；在 G_2 里平均进行 9.4 轮信息传播扩散，平均最大传播范围是 484.4 个节点。

再次随机生成 500 个样本，此时控制种子集规模从 1 开始递增，得到的相关数据 (进行多次实验取平均值) 如下表所示：

种子集规模	1	2	3	4	5
总传播范围	443	482	491	495	500
G_1 中传播轮次	2	7	11	9	7
G_2 中传播轮次	2	4	2	2	4
G_1 中传播范围	440	471	489	494	498
G_2 中传播范围	412	457	465	470	475

表 4.2　500 个随机样本控制种子集规模相关结果

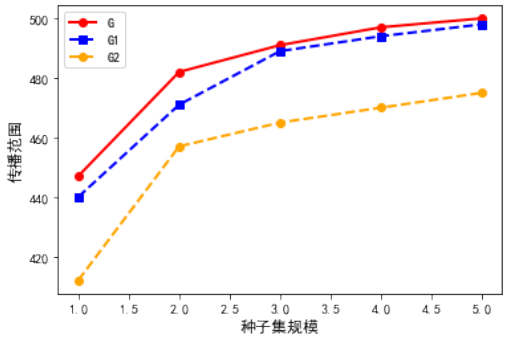


图 4.4　500 个随机样本（控制种子集规模）信息扩散范围示意图

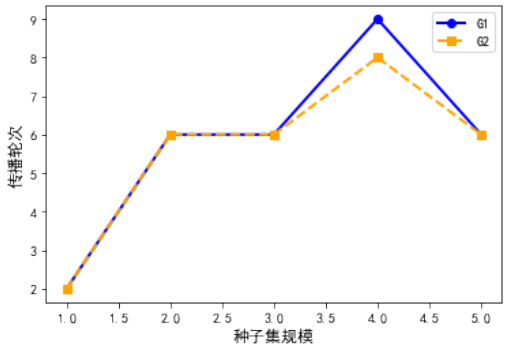


图 4.5　500 个随机样本（控制种子集规模）信息扩散轮次示意图

通过图 4.5 可以看出，随着种子集合大小的增加，在 G_1 和 G_2 中信息扩散的轮次不一定增加，这是因为随着种子节点的增加，原先可能需要在 $n+1$ 轮扩散才有几率被激活的节点现在有可能在第 n 轮扩散就被新加入的种子节点激活，因此总的扩散轮次有可能反而减少。

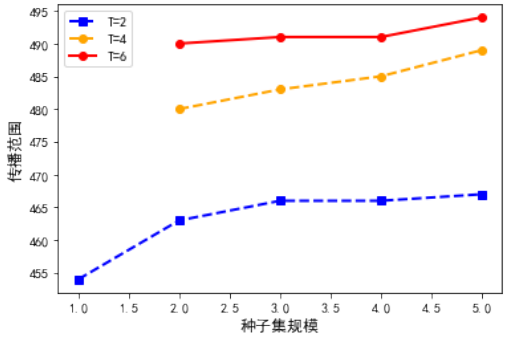


图 4.6　500 个随机样本控制种子集规模得到不同轮次传播范围示意图

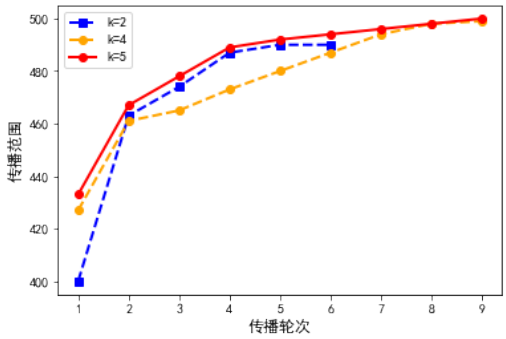


图 4.7　500 个随机样本控制轮次不同种子集规模传播范围示意图 (k 代表种子集规模)

通过以上结果，我们可以发现在种子集加入新的节点时，总传播范围虽然增加，但是增加量是下降的，这符合边际效应以及前文提到的“次模性”的特点。

通过第三章设计实验我们可以知道，节点的影响力的设定可以很大程度上影响传播范围、传播轮次和将所有节点激活需要的种子集规模。采取不同比例混合节点的度中心性和介数中心性，得到不同的节点影响力，分析节点影响力中位数和其对影响力最大化问题的影响如下表所示：

节点影响力中位数：	0.2	0.3	0.4	0.5	0.6	0.7	0.8
激活所有节点需要的种子集规模	305	128	15	5	3	2	2

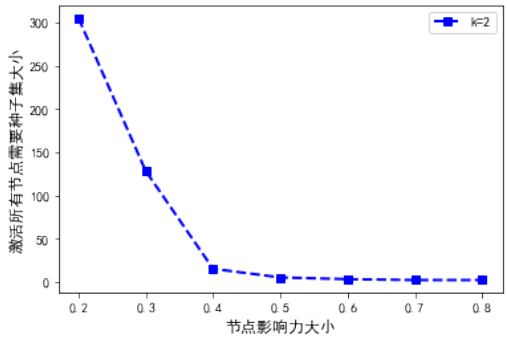


图 4.8　500 个随机样本控制节点影响力示意图

由上图可以简单看出，随着节点影响力中位数的增大，激活所有节点所需要的种子集规模在减小。通过分析，发现随着节点影响力中位数的增大，其对于激活所有节点所需要的种子集规模的边际效应递减。这是由于独立级联模型是一个概率传播模型，在这个传播模型内，当每个节点的影响力很小时，处于未激活态的节点很难被邻居节点激活；而当每个节点的影响力较大时，任意一个处于活跃态的节点都可以用这个较大的概率来尝试激活处于未激活态的邻居节点，则所有节点都被激活的概率得到相对大幅提高。

4.5　本章小结

本章首先按照制定概率（数据来源于第七次人口普查）随机生成 20 个样本进行预实验，确保假设的实验可以正常进行；随后用同样方式随机生成 500 个样本，不限制种子集规模，得到一系列数据进行展示与分析；最后再次随机生成 500 个样本，控制种子集规模从 1 开始扩大，得到一系列数据进行展示与分析。

5 总结与展望

本文的主要工作是基于对现实情况的观察理解，选择以“强关系”社交网络与“弱关系”社交网络来组成双层社交网络，并将独立级联模型作为信息扩散传播的模型。

该网络的建立能很好的反映线下与线上社交网络的特点，是后续研究探讨的至关重要的理论基础。

理论上，本文介绍了社交网络的现实背景、国内外发展现状；简单介绍了社交网络的一些基础概念，如节点、影响力最大化、次模性；介绍了两种基本的信息扩散模型——线性阈值模型与独立级联模型；为了解决影响力最大化问题而引入了贪心算法；为每个节点的影响力大小作出了评估度量；介绍了多种网络模型，并根据实际建立起双层社交网络模型。

实践上，本文采用 2021 年全国第七次人口普查的数据进行模拟，通过 python 对理论进行实践，将结果以图形形式展示出来并进行分析。

当然，本文也有一定的不足，比如为了简化模型，在构建“强关系”社交网络时，粗略根据“差异度分布直方图”选取“差异度”小于 0.4 即可建立连接，并未对“差异度”进行更多数据分析；在通过 G_1 映射形成初步的 G_2 时，映射的概率直接选取 0.5；在衡量节点影响力大小时，采用“度中心性”和“介数中心性”归一化后分别乘以 0.6、0.4 并混合作为每个节点的影响力大小，这样衡量节点影响力是否符合实际还有待考究，并且混合比例也需要根据数据情况改进等等。在将来的研究中，应该采取更符合统计学规律的方式来确定这些参数。

参考文献

[1] KEMPE D, KLEINBERG J, TARDOS É. Maximizing the spread of influence through a social network[C]//Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and Data Mining. 2003: 137-146.

[2] DOMINGOS P, RICHARDSON M. Mining the network value of customers[C]// Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and Data Mining. 2001: 57-66.

[3] RICHARDSON M, DOMINGOS P. Mining knowledge-sharing sites for viral marketing[C]//Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and Data Mining. 2002: 61-70.

[4] LESKOVEC J, KRAUSE A, GUESTRIN C, et al. Cost-effective outbreak detection in networks[C]//Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and Data Mining. 2007: 420-429.

[5] KUMAR S, GUPTA A, KHATRI I. Csr: A community based spreaders ranking algorithm for influence maximization in social networks[J]. World wide web, 2022, 25(6): 2303-2322.

[6] ZHANG J X, CHEN D B, DONG Q, et al. Identifying a set of influential spreaders in complex networks[J]. Scientific reports, 2016, 6(1): 27823.

[7] 胡斌, 王敬志, 刘鹏. 基于双层网络的混合 PSO 算法的 RBF 建模[J]. 西南科技大学学报, 2011, 26(2): 78-81.

[8] WANG C, CHEN W, WANG Y. Scalable influence maximization for independent cascade model in large-scale social networks[J]. Data Mining and Knowledge discovery, 2012, 25: 545-576.

[9] GAO S, MA J, CHEN Z, et al. Ranking the spreading ability of nodes in complex networks based on local structure[J]. Physica A: Statistical Mechanics and its Applications, 2014, 403: 130-147.

[10] BAVELAS A, BEAUGUITTE L, FEN-CHONG J. A. bavelas, 1950, communication patterns in task-oriented groups. version bilingue et commentée[J]. 2021.

[11] FREEMAN L C. A set of measures of centrality based on betweenness[J]. Sociometry, 1977: 35-41.

[12] MARCHIORI M, LATORA V. Harmony in the small-world[J]. Physica A: Statistical Mechanics and its Applications, 2000, 285(3-4): 539-546.

[13] LEI S, MANIU S, MO L, et al. Online influence maximization[C]//Proceedings of the 21th ACM SIGKDD international conference on Knowledge discovery and Data Mining. 2015: 645-654.

[14] WEN Z, KVETON B, VALKO M, et al. Online influence maximization under independent cascade model with semi-bandit feedback[J]. Advances in neural information processing systems, 2017, 30.

[15] ERDŐS P, RÉNYI A, et al. On the evolution of random graphs[J]. Publ. math. inst. hung. acad. sci, 1960, 5(1): 17-60.

[16] WATTS D J, STROGATZ S H. Collective dynamics of ‘small-world’ networks [J]. nature, 1998, 393(6684): 440-442.

[17] BARABÁSI A L, ALBERT R. Emergence of scaling in random networks[J]. science, 1999, 286(5439): 509-512.

[18] SHANNON C E. A mathematical theory of communication[J]. The Bell system technical journal, 1948, 27(3): 379-423.

[19] 胡海波, 徐玲, 王科, 等. 大型在线社会网络结构分析[J]. 上海交通大学学报, 2009(4): 587-591.

[20] 韩忠明, 陈炎, 刘雯, 等. 社会网络节点影响力分析研究[J]. 软件学报, 2016, 28 (1): 84-104.

[21] 宁吉. 第七次全国人口普查主要数据情况[J]. 中国统计, 2021, 5: 4-5.

致谢

本文于 2024 年 2 月正式动笔，于 2024 年 4 月完稿，期间发生一系列的意外，幸运的是得到了多方面的帮助，论文能够按时完成，我在此对所有帮助过我的人表达由衷的感谢。

首先，我需要感谢的是我的亲人。他们并不懂得我学的知识和我做的事情，但是他们依然表现出无限的理解包容与支持，让我在充满爱的环境里学习、完成论文，并且他们积极地为我的未来提供宝贵的建议，我在此表示感谢。这几年对于我的家庭来说充满了坎坷，好在我们一家人拧成一股绳，一起面对每一个困难，一步步走到今天，我对我的所有家人表示感谢。

我的导师——胡捷导师，从选题开始便为我尽心尽力地指导，并以其严谨求实的治学态度、高度的敬业精神和大胆创新的进取精神引领我的撰写工作。在论文撰写的进程中，胡捷导师经常提醒我各种进程的时间节点，牢牢把握我的论文完成进度，并为我的论文提供宝贵的修改意见，是我完成本文的重要助力者，在此，特别对胡捷导师表示感谢。

在撰写毕业论文的过程中，我的三位舍友——唐振哲、邱嘉诚、郑霖恺同学互相鼓励，互相监督；在大学四年里，每当我学习知识遇到难点时（包括毕业论文的相关知识学习），他们愿意一起帮助我思考、理解，攻克难关，是我学习成长路上的伙伴。记得初入大学时，思维方式难以从高中转换过来的我在学习上十分吃力，在此感谢我的舍友和陈越学长，不厌其烦地为我解答疑惑，让我学习成绩一步步好起来，我很难想象没有他们我该如何进行大学的学习。

感谢来自珞之闽 (武汉大学福建老乡会) 的朋友们，感谢你们的陪伴让我在大学四年的时光里充满快乐。

在此离别之际，我献上最真诚的感谢和最真挚的祝福，祝愿我们无论走上哪条人生道路，都能得偿所愿，前途无量。

最后，再次对关心、帮助我的所有人表示衷心地感谢。

附录 A 数据

A.1 预实验相关数据

$$\begin{bmatrix}
0 & 0.5 & 0 & 0 & 1 & 0 \\
0 & 1 & 0.5 & 0 & 1 & 1 \\
0 & 1 & 0.5 & 0 & 0.5 & 0 \\
1 & 0.5 & 1 & 0 & 0.125 & 0 \\
1 & 0.5 & 0.5 & 0 & 0 & 0 \\
0 & 0.5 & 0.5 & 1 & 0.5 & 0 \\
0 & 0.5 & 0 & 0 & 0 & 1 \\
1 & 0.5 & 0.5 & 0 & 0.5 & 0 \\
0 & 0.5 & 0.5 & 0 & 1 & 1 \\
1 & 0 & 10 & 0.5 & 0 & \\
0 & 1 & 0 & 0 & 0.75 & 1 \\
0 & 0 & 0 & 0 & 0.5 & 0 \\
1 & 0.5 & 0.5 & 0 & 0.875 & 0 \\
1 & 0 & 0.5 & 0 & 0 & 1 \\
0 & 0.5 & 0.5 & 0 & 0.5 & 1 \\
0 & 0.5 & 0.5 & 0 & 0.875 & 1 \\
1 & 0.5 & 0 & 0 & 0.5 & 0 \\
0 & 1 & 0.5 & 0 & 0.125 & 0 \\
1 & 1 & 0.5 & 0 & 0.875 & 1
\end{bmatrix} \tag{A.1}$$

A.2 实验代码

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import math
import random
import networkx as nx
import copy
##进行熵权法为指标赋值，这部分不做修改
e1=-(0.5124*math.log(0.5124)+0.4876*math.log(0.4876))/math.log(2)
e2=-(0.1795*math.log(0.1795)+0.6335*math.log(0.6335)+0.187*math.log(0.187))/math.log(3)
#p_e3_1=0.1795*0.0270+0.6335*0.0550+0.1870*0.4760
#p_e3_2=0.1795*0.9690+0.6335*0.7375+0.1870*0.4630 计算e3需要的概率
#p_e3_3=0.1795*0.0040+0.6335*0.1585+0.1870*0.0400
```

```
#p_e3_4=0.1795*0.00+0.6335*0.0490+0.1870*0.0210
e3=-(0.1287*math.log(0.1287)+0.7277*math.log(0.7277)+0.1086*math.log(0.1086)+0.0349*math.log(
e4=-(0.9111*math.log(0.9111)+0.0889*math.log(0.0889))/math.log(2)
e5=-(0.2286*math.log(0.2286)+0.1104*math.log(0.1104)+0.0023*math.log(0.0023)+0.1177*math.log(
e6=-(0.6389*math.log(0.6389)+0.3611*math.log(0.3611))/math.log(2)
d1=1-e1
d2=1-e2
d3=1-e3
d4=1-e4
d5=1-e5
d6=1-e6
sum_d=d1+d2+d3+d4+d5+d6
lambda1=d1/sum_d
lambda2=d2/sum_d
lambda3=d3/sum_d
lambda4=d4/sum_d
lambda5=d5/sum_d
lambda6=d6/sum_d##熵权法结束
##定义根据制定概率生成制定数据
def number_of_certain_probability(sequence, probability):
    x = random.uniform(0, 1)
    cumulative_probability = 0.0
    for item, item_probability in zip(sequence, probability):
        cumulative_probability += item_probability
        if x < cumulative_probability:
            break
    return item
##生成特定指标（单一指标）的样本（已进行归一化）
def sample_making(size,sequence,probability):
    N_sample_making=[]
    for i in range(size):##生成size个样本的某一个属性
        t=number_of_certain_probability(sequence,probability)
        N_sample_making.append(t)
        N_sample_making_min=min(N_sample_making)
        N_sample_making_max=max(N_sample_making)
    for i in range(size):
        N_sample_making[i]=(N_sample_making[i]-N_sample_making_min)/(N_sample_making_max-N_sam
    return N_sample_making

x_give_values=[[1,2],[1,2,3],[1,2,3,4],[1,2],[1,2,3,4,5,6,7,8,9],[1,2]]
p_sample_making=[[0.5124,0.4876],[0.1795,0.6335,0.1870],[0.1287,0.7277,0.1086,0.0350],[0.9111
##创造20个样本
N=[]
```

```
for i in range(6):
    N.append(sample_making(500,x_give_values[i],p_sample_making[i]))
N=np.array(N)
N=N.T
def chayidu(m,n):##差异度函数
    R1=0;
    R=0;
    for i in range(6):
        R1 += lambda_list[i]*((N[m,i]-N[n,i])**2)
        R = math.sqrt(R1)
        R = int(R*1000)/1000
    return R
lambda_list=[lambda1,lambda2,lambda3,lambda4,lambda5,lambda6]##权重
def chayidujuzhen(size):##差异度矩阵函数
    r=np.zeros((size,size))##初始化差异度矩阵
    for i in range(size): ##获得差异度矩阵
        for j in range(size):
            r[i,j]=chayidu(i,j)
    return r
##获得差异数矩阵
r=chayidujuzhen(len(N))
##绘制差异度分布直方图
RR=[]
for i in range(len(N)):
    for j in range(i+1,len(N)):
        RR.append(r[i,j])

#设置绘图风格
plt.style.use('ggplot')
#处理中文乱码
plt.rcParams['font.sans-serif'] = ['Microsoft YaHei']
#坐标轴负号的处理
plt.rcParams['axes.unicode_minus']=False
# 绘制直方图
plt.hist(x = RR, # 指定绘图数据
         bins = 40, # 指定直方图中条块的个数
         color = 'steelblue', # 指定直方图的填充色
         edgecolor = 'black' # 指定直方图的边框色
         )
# 添加x轴和y轴标签
plt.xlabel('差异度')
plt.ylabel('频数')
# 添加标题
```

```
plt.title('差异度分布')
# 显示图形
plt.show()
##建立强关系G1散点图
G1 = nx.Graph()
for i in range(len(N)):
    G1.add_node(i+1)
edge_count_G1=0
for i in range(len(N)):
    for j in range(i+1,len(N)):
        if (r[i,j]<=0.4):##定义差异度低于多少可以连接
            G1.add_edge(i+1, j+1)
            edge_count_G1+=1

plt.figure(figsize=(10, 10))


degree_G1=dict(G1.degree)




G2 = nx.Graph()
for i in range(len(N)):
    G2.add_node(i+1)
edge_count_G2_1=0
for i in range(len(N)):
    for j in range(i+1,len(N)):
        if (G1.has_edge(i+1,j+1)):
            suijishu1=random.random()
            if(suijishu1 <= 0.5):##映射概率
                G2.add_edge(i+1,j+1)
                edge_count_G2_1+=1




degree_G2=dict(G2.degree)
sum_degree_G2 = 0

for i in range(len(N)):
    sum_degree_G2 += G2.degree(i+1)


edge_count_G2_2=0
```

```
143
144 for i in range(len(N)):
145     if (G2.degree(i+1) == 0):
146         while(G2.degree(i+1) == 0):
147             for j in range(len(N)):
148                 if (G2.degree(j+1) != 0):
149                     suijishu_G2=random.random()
150                     if(suijishu_G2 <= (G2.degree(j+1)/sum_degree_G2)):
151                         if(i!=j):
152                             G2.add_edge(i+1,j+1)
153                             edge_count_G2_2 += 1
154
155 edge_count_G2=edge_count_G2_1+edge_count_G2_2
156
157
158
159
160 degree_G2=dict(G2.degree)
161
162 def power_measure(G,k1,k2):##计算节点影响力，采用介数中心性和度中心性混合评估
163     betweenness_G =
        nx.betweenness_centrality(G,normalized=False)##计算介数中心性
164     betweenness_G=list(betweenness_G.values())
165     max_betweenness_G=max(betweenness_G)
166     min_betweenness_G=min(betweenness_G)
167
168     for i in range(len(N)):
169         betweenness_G[i]=(betweenness_G[i]-min_betweenness_G)/(max_betweenness_G-min_betweenne
170
171     degree_centrality_G = nx.degree_centrality(G)##计算度中心性
172     degree_centrality_G=list(degree_centrality_G.values())
173
174     max_degree_centrality_G=max(degree_centrality_G)
175     min_degree_centrality_G=min(degree_centrality_G)
176
177     for i in range(len(N)):
178         degree_centrality_G[i]=(degree_centrality_G[i]-min_degree_centrality_G)/(max_degree_ce
            -min_degree_centrality_G)
179
180     mix_G_P=[]
181     for i in range(len(N)):
182         mix_G_P.append(betweenness_G[i]*k1+degree_centrality_G[i]*k2)
183     return mix_G_P
```

```
184
185 power_G1=power_measure(G1,0.4,0.2) ##计算G1，G2图节点影响力
186 power_G2=power_measure(G2,0.4,0.2)
187
188 G1_matrix = nx.to_numpy_matrix(G1)##计算
189 G2_matrix = nx.to_numpy_matrix(G2)
190
191 # IC模型
192 # 输入：图的邻接矩阵、初始节点集S
193 # 输出：感染点的个数
194 def ICModel(data_1,data_2,init_set,power_1,power_2):
195     # 活集A，存储本轮被激活的所有点
196     # 初始值为S
197     active_set = init_set
198     round_count_1=0
199     round_count_2=0
200     active_set_num_each_round_1=[]
201     active_set_num_each_round_2=[]
202     # 所有节点的激活状态
203     # 0为未激活，1为激活
204     active_status_set = np.zeros(len(data_1), dtype=int)
205     active_status_set_1 = np.zeros(len(data_1), dtype=int)
206     active_status_set_2 = np.zeros(len(data_2), dtype=int)
207
208     # 激活所有起始点
209     for index2 in range(len(init_set)):
210         active_status_set[init_set[index2]] = 1
211         active_status_set_1[init_set[index2]] = 1
212         active_status_set_2[init_set[index2]] = 1
213     # 开始循环
214     while active_status_set.sum() < len(data_1):
215
216         # A的非活邻居集N
217         # 存储本轮可能被传染的点
218         neighbor_set_1 = []
219         neighbor_set_2 = []
220         # 取出未激活可达点，及点的所有有效边的概率
221         p_dictionary_1 = {}
222         p_dictionary_2 = {}
223         for d_index in range(len(data_1)):
224             for s_index in range(len(active_set)):
225                 # 可达且未被激活
226                 if data_1[active_set[s_index], d_index] != 0.0 and
```

```
                active_status_set[d_index] != 1:
227             if d_index not in neighbor_set_1:
228                 neighbor_set_1.append(d_index)
229                 p_dictionary_1[d_index] = []
230             p_dictionary_1[d_index].append(power_1[active_set[s_index]])
231
232         if data_2[active_set[s_index], d_index] != 0.0 and
                active_status_set[d_index] != 1:
233             if d_index not in neighbor_set_2:
234                 neighbor_set_2.append(d_index)
235                 p_dictionary_2[d_index] = []
236             p_dictionary_2[d_index].append(power_2[active_set[s_index]])
237
238
239
240
241
242
243
244
245     # 出口：不存在未激活可达点
246      if len(neighbor_set_1) == 0 and len(neighbor_set_2) == 0:
247         break
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266         # 计算未激活可达点的激活概率
267      neighbor_p_set_1 = []
```

```
        neighbor_p_set_2 = []
        for index2 in range(len(neighbor_set_1)):
            p_one_1 = p_dictionary_1[neighbor_set_1[index2]]
            p_1 = 1.0
            for i in range(len(p_one_1)):
                p_1 = p_1 * (1 - p_one_1[i])
            p_1 = 1 - p_1
            neighbor_p_set_1.append(p_1)

        for index2 in range(len(neighbor_set_2)):
            p_one_2 = p_dictionary_2[neighbor_set_2[index2]]
            p_2 = 1.0
            for i in range(len(p_one_2)):
                p_2 = p_2 * (1 - p_one_2[i])
            p_2 = 1 - p_2
            neighbor_p_set_2.append(p_2)









        # 清空A集
        active_set = []
        active_set_1 = []
        active_set_2 = []
```

```
311         # 激活
312         for index3 in range(len(neighbor_set_1)):
313             seed = random.random()
314             if seed < neighbor_p_set_1[index3]:
315                 active_status_set_1[neighbor_set_1[index3]] = 1
316                 active_set_1.append(neighbor_set_1[index3])
317 
318         if range(len(neighbor_set_1)):
319             round_count_1+=1
320             active_set_num_each_round_1.append(active_status_set_1.sum())
321 
322 
323         for index3 in range(len(neighbor_set_2)):
324             seed = random.random()
325             if seed < neighbor_p_set_2[index3]:
326                 active_status_set_2[neighbor_set_2[index3]] = 1
327                 active_set_2.append(neighbor_set_2[index3])
328 
329         if range(len(neighbor_set_2)):
330             round_count_2+=1
331             active_set_num_each_round_2.append(active_status_set_2.sum())
332 
333 
334 
335         active_set= list(set(active_set_1) | set(active_set_2))
336         active_status_set=np.where(active_status_set_1>active_status_set_2,active_status_set_1
337 
338 
339     # print(init_set, active_status_set)
340     return
        active_status_set,active_status_set_1,active_status_set_2,round_count_1,round_count_2,
341 
342     init_set = []
343     init_set_print = []
344     # 激活节点个数
345     spread_num = 0
346     active_set_num_each_round_G1_final=[]
347     active_set_num_each_round_G2_final=[]
348     active_status_total_set=[]
349     round_in_G1_final=0
350     round_in_G2_final=0
351     size_of_seed_set=10000##控制种子集大小
352     active_status_total_set_final=[]
```

```
353     t_control=1000000##控制每次ic的轮次
354
355     # （贪心算法）往S中加入新节点，使得每次加入后激活的节点数最多
356     # 当所有节点都被激活时停止循环
357     # f = open('result.txt', 'a')
358     while spread_num != len(G1_matrix):
359         max_spread_num = 0
360         max_spread_num_G1=0
361         for index in range(len(G1_matrix)):
362             if index not in init_set:
363                 test_S = copy.deepcopy(init_set)
364                 test_S.append(index)
365
366
367                 active_status_total_set,active_status_set_G1,active_status_set_G2,round_in_G1,r
                        ICModel(G1_matrix,G2_matrix,test_S,power_G1,power_G2)
368
369                 result_spread_num = active_status_total_set.sum()
370
371
372                 if result_spread_num > max_spread_num:
373                     active_status_total_set_final=list(active_status_total_set)
374                     max_spread_num = result_spread_num
375                     max_spread_node = index
376                     max_spread_num_G1=active_status_set_G1.sum()
377                     round_in_G1_final=round_in_G1
378                     round_in_G2_final=round_in_G2
379                     active_set_num_each_round_G1_final=active_set_num_each_round_G1
380                     active_set_num_each_round_G2_final=active_set_num_each_round_G2
381
382
383                 if result_spread_num == max_spread_num:
384                     if active_status_set_G1.sum()>max_spread_num_G1:
385                         max_spread_num_G1=active_status_set_G1.sum()
386                         max_spread_node = index
387                         active_status_total_set_final=list(active_status_total_set)
388                         round_in_G1_final=round_in_G1
389                         round_in_G2_final=round_in_G2
390                         active_set_num_each_round_G1_final=active_set_num_each_round_G1
391                         active_set_num_each_round_G2_final=active_set_num_each_round_G2
392
393         spread_num = max_spread_num
394         init_set.append(max_spread_node)
```

```
395
396         init_set_print.append(max_spread_node+1)
397
398
399         print("init_num: ", len(init_set), " spread_num: ", max_spread_num, "
                S:",
                init_set_print,"G1中扩散轮次:",round_in_G1_final,"G1每轮激活节点数: ",
                active_set_num_each_round_G1_final,"G2中扩散轮次: ",round_in_G2_final,"G2每轮激活节
400     ##  print("激活态节点分别为: ")
401     ##  for i in range(len(active_status_total_set_final)):
            ##可以输出所有激活态的节点
402     ##      if active_status_total_set_final[i]==1:
403     ##          print(i+1)
404               ##控制种子集大小
405         if len(init_set)==size_of_seed_set:
406             break
407         # f.write("init_num: " + str(len(init_set)) + " spread_num: " +
                str(max_spread_num) + " S:" + str(init_set) + "\n")
408
409     # f.close()
```