基于复杂网络视角探究社交网络上的信息传播及其衍生现象

摘要

本文针对社交网络上的信息传播、用户观点中立共识和观点极化现象、尖叫效应、回声室效应、信息茧房等现象的形成机制和影响因素展开研究，运用了 SIR 病毒传播动力学理论、复杂网络、观点动力学、自发对称性破缺等理论，并借助网络爬虫、自然语言处理等技术，构建了 SIR-BASFN 模型、PageRank-SIR-RUCM 模型、观点极化与中立共识产生机制模型、回声室效应产生机制模型、信息茧房形成机制等模型，综合运用了 python、MATLAB 等编程软件进行求解，得出了网络中度较大的节点作初始感染节点与话题固有传播概率较大时对话题传播的影响较为显著、观点极化与中立共识与用户之间的信任阈值与趋同系数密切相关、尖叫效应往往是由于网络中话题发布者的度较大，且发布的话题的固有传播概率较大引起的、回声室效应是由于网络结构发生改变，产生相对独立的团簇，即发生自发对称性破缺引起的、信息茧房是由于社区之间的信任阈值（用户心理偏好）和平台推荐算法以及用户间的相互影响等综合因素导致，同时我们也对上述现象的影响因素进行定性或定量的分析，最后给出了破除尖叫效应和回声室效应以及规避信息建房的策略，并从自底向上地分析了破除信息茧房地解决方案。

针对问题一，要定量描述社交媒体上的话题传播及其影响因素，我们循序渐进地提出了 PageRank-SIR-RUCM 模型，运用 MATLAB 求解得出影响话题传播主要因素；同时我们也设计爬虫针对微博爬取观点极化与中立共识话题相关数据，借助自然语言处理分析用户对话题的态度，结合数据分析分别得到一个观点极化的话题和一个中立共识的话题。

针对问题二，我们运用观点动力学成功解释了观点极化与中立共识以及信息茧房的形成机制；同时我们也发现 PageRank-SIR-RUCM 模型也能很好的解释观点极化与中立共识，并且承接问题一的分析结论可以直接得出尖叫效应的产生机制；除此之外，我们从物理学角度建立自发对称性破缺模型解释了回声室效应的形成机制，并提出（T, T^N）相图用于简洁明了地呈现回声室，同时也对上述现象地影响因素建立相应模型进行分析，得到重要结论以解决问题三和问题四。

针对问题三，我们基于问题一和问题二地综合结果提出破除尖叫效应和回声室效应、规避信息茧房的策略。

针对问题四，我们结合问题二的结论以及通过文献调研，自底向上地提出了破除信息茧房的合理建议。

在文末我们也对模型的优缺点进行了客观的评价，提出模型有待改进之处。

关键字： 社交网络　话题传播　复杂网络　观点动力学　MATLAB

目录

一、 问题重述

1.1 背景知识

“尖叫效应”[1] 是心理学中的一个著名效应。例如在一个人潮涌动的公众场合，如果有人突然歇斯底里地尖叫，往往能快速吸引人们的注意力并博取眼球。在网络信息传播中，“尖叫效应”也无处不在。一些网络平台利用大数据和人工智能，获取并分析用户浏览记录和兴趣爱好等信息，大量推送段子、恶搞、色情等低俗内容。无论是从满足人们的猎奇心理，还是引发人们的指责批评，传播者都能从中获取高额的流量和点击率。

“回声室效应”[2, 3, 4, 5, 6] 指的是在一个相对封闭的媒体环境中，一些意见相近的声音不断重复，甚至夸张扭曲，令处于其中的大多数人认为这些声音就是事实的全部，不知不觉中窄化自己的眼界和理解，走向故步自封甚至偏执极化。在现代社会中，由于互联网以及社交媒体的发展，在网络信息传播中“回声室效应”愈发明显。部分商业网站会分析记录用户的搜寻结果以及使用习惯，持续地将一位用户所喜欢的内容提供给该用户，导致一个人在同一网站中接受到的资讯被局限于某个范围内。

“尖叫效应”与“回声室效应”容易导致“信息茧房”[1, 7, 8] 的形成。所谓“信息茧房”指的是，在信息传播中人们自身的信息需求并非全方位的，只会选择自己想要的或能使自己愉悦的信息，久而久之接触的信息就越来越局限，最终将自己桎梏于像蚕茧一般的“茧房”中，失去对其他不同信息的了解能力和接触机会。

在全新的信息传播格局下，如何破除“尖叫效应”与“回声室效应”，走出“信息茧房”，是当前迫切需要解决的现实问题，即如何从信息传输的顶层设计、推荐算法的公平性和广大网络用户的责任担当等方面，帮助公众对新闻事件乃至社会现实有一个相对准确、清晰的认识和判断，并在主流意识和个性化信息之间找到平衡点，使得网络舆论环境更具理性和建设性。

1.2 具体问题

1. 针对某些话题，在微信、微博、Facebook 和 Twitter 等社交媒体上下载相关数据，定量描述该话题（或信息）的传播过程，并分析其影响因素。该数据分析需至少针对两种不同的话题展开讨论，其中一个话题最终观点趋于相同（中立共识），另一话题最终观点趋于两极分化（观点极化）。

2. 建立数学模型刻画中立共识和观点极化的产生机制，探索“尖叫效应”、“回声室效应”与“信息茧房”的形成机制，并讨论话题的吸引度、用户的活跃度、用户心理、不同用户间的相互影响、平台推荐算法等因素对形成这些现象的影响。

3. 根据问题 2 建立的数学模型，制定破除“尖叫效应”和“回声室效应”、规避“信息茧房”的策略。

4. 基于上述数据分析与数学模型，针对如何破除“信息茧房”撰写 1-2 页报告，分别对

政府的顶层设计、主流媒体的引领和广大网络用户的责任担当提出相应的解决方案或建议。

1.3 相关数据

本文所有真实数据均来源于微博，我们建立了一个开源仓库用于储存本文的所有代码以及数据，可以访问我们的深圳杯代码 + 数据 GitHub 仓库

二、 模型的假设

- 假设社交网络上的话题传播过程服从 SIR 模型
- 假设社交网络服从 BASFN 模型
- 假设本文使用的网络模型 BASFN 是非度关联的，即对于1中的条件概率 $P(k'|k) = \frac{k'P(k')}{\langle k \rangle}$（$\langle k \rangle$ 为网络的平均度）
- 假设社交网络在形成过程中，所有节点都会倾向于与其观点相近的节点相连接，并遵循节点出度越大，与新节点连接的概率越大
- 假设 PageRank 值可以表示节点的权威度
- 假设用户在线概率为常数
- 假设用户上网浏览网页时，用户选择下一个页面的过程与过去浏览哪些页面无关
- 假设观点 x_i 的参数函数 $\rho(x)$ 满足周期性边界条件式24

三、 名词解释与符号说明

3.1 名词解释

1. 度：网络中的节点连接的边的数目。直观地看，一个节点的度越大，这个节点在某特定的意义上就越重要。
2. 度分布：网络中节点的分布情况，用函数 $P(k)$ 来表示，描述网络中度为 k 的节点在整个网络节点数的占比，也可以理解为在网络中随机抽取到度为 k 的节点的概率为 $P(k)$。
3. 认知偏爱：指热门倾向于以某种方式获取和处理新信息，已确认自己的先入之见，避免与先前的信念矛盾。
4. 社会影响：指一个人的情绪、观点或行为受到他人影响的过程。
5. 自发对称性破缺 (Spontaneous symmetry breaking)：物理学中的概念，指出现一种不对称状态，由于习通原本的对称性自发破缺而稳定。

3.2 符号说明

本文中所用到的符号说明见表 1, 表中未标出的符号也会在后文有相应说明。同时，这里也提醒有部分符号会有重复使用但意义不同的情况，具体细节文章中会说明，请仔细阅读，小心混淆。

四、 问题分析

问题 1：为定量描述社交媒体上的话题传播过程，我们需要分析话题从发起到传播结束的全过程。因此，需要建立社交网络模型来抽象的描述网络结构，抽象出影响话题传播的主要因素，同时需要构建描述话题传播的动力学机制的模型。同时，为直观地呈现话题传播过程，我们需要对话题传播模型进行模拟以揭示对话题传播过程有重要影响的因素。针对两种类型的话题（中立共识和观点极化）的数据分析，我们需要借助爬虫爬取相应话题的阅读量、点赞数、转发数、评论数、评论的态度等数据以分析话题传播过程中的细节和影响话题传播的因素。

问题 2：对于探讨中立共识和观点极化的机制，我们需要理解网络中观点变化的机制，建立观点动力学模型，通过模拟揭示对形成两种现象有重要影响的参数以阐明其产生机制；对于尖叫效应，我们需要理解尖叫效应，同时再问题一模型的基础上定义描述尖叫效应的数学定义，通过模拟计算来阐述其发生机制；对于回声室效应，我们需要认识回声室效应的本质是有着共同信念和偏好的人形成相对封闭网络结构，在该网络结构中的人的观点、偏好往往趋于一致，他们的信念会因为志同道合而得到加强，同时有可能形成信息茧房，因而我们需要建立相应模型来表示该过程的发生机制；对于信息茧房形成机制，我们首先要理解回声室效应与信息茧房的关联，同时建立相应的模型描述该现象的产生机制；而对于上述话题传播的衍生现象（观点计划、中立共识、尖叫效应、回音室效应、信息茧房）的影响因素分析则需要将各主要影响因素进行数学抽象简化，并归纳进这些现象的产生机制模型采用控制变量法做模拟以定性分析这些影响因素的作用。

问题 3：结合问题 2 中的尖叫效应、回声室效应、信息茧房产生机制于影响因素分析模型提出合理的策略。

问题 4：结合问题 2 中的信息茧房影响因素分析明确对形成茧房有显著影响的因素，同时结合文献调研法提出合理的决策和建议。

问题求解的总体思路如图1.

表 **1**　本文所用符号的相关说明与解释

符号	意义
$n_{k,S}(t)$、$n_{k,I}(t)$、$n_{k,R}(t)$	分别对应 t 时刻度为 k 的节点中处于 S 态、I 态、R 态的节点数
N	节点、用户、网页等的总数
k	网络中节点的度
ρ_S、ρ_R	分别对应 S 态、R 态的节点密度
P(k)	网络的度分布
θ	用户在线概率
$\eta(i)$	用户感染概率
$\alpha(t)$	用户对话题的免疫概率
$\delta(i)$	用户直接从 S 态转化为 R 态的概率
x_i	用户对话题的观点
$\rho(x)$	x 所满足的周期性边界条件函数
ε、δ	均代表信任阈值、容限距离
μ、ζ	均指收敛系数、趋同系数
$G(V,E)$	有向图，V 代表节点，E 代表边
$\otimes$	一种名为 Kronecker product 的运算符
C_i	用户 i 的话题集合
T_i、T_i^N	分别对应用户话题倾向平均值和用户邻居话题倾向平均值
$o_i(t)$	用户 i 在 t 时刻的意见值
d_{ij}	用户 i 和用户 j 之间的意见距离
w	影响因素的权重

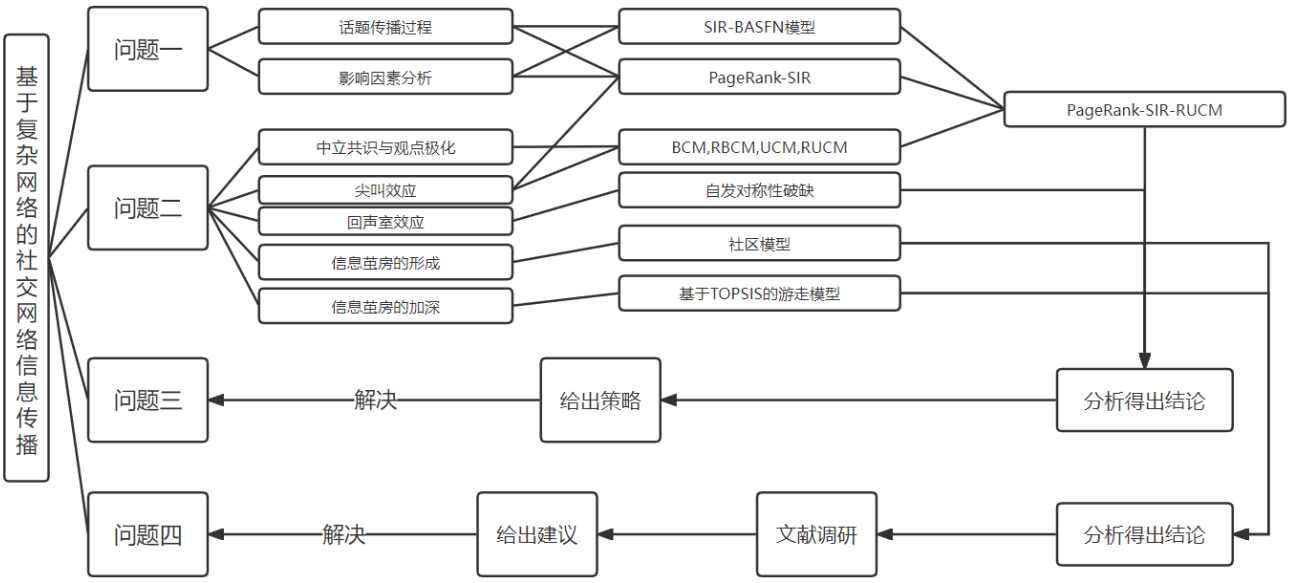


图 **1**　问题求解的总体思路框图

五、 模型的建立与求解

5.1 问题一分析与求解

5.1.1 SIR-BASFN 模型

已有研究表明对于社交媒体上的话题传播的动力学行为服从 SIR 模型 [9, 10]，我们将基于 SIR 模型讨论由社交媒体用户构成的网络（以下简称网络）上的话题传播过程。下面定义网络节点（人群）的类型：

- S 态：没有听过话题的节点（Ignorants），类似于病毒传播模型中易感染的节点（Susceptile），对应于不知道话题的个体。
- I 态：话题传播节点（Spreaders），类似于病毒传播模型中的易感染节点（Infected），对应于知道话题并有能力继续传播话题的个体。
- R 态：听到话题并不传播话题的节点（Stiflers），类似于病毒传播模型中的免疫节点（Remove），对应于知道话题但不传播话题的个体。

假设网络上有 N 个节点，每个节点代表一个可传播消息的个体，它们传播消息的行为方式如下 [11]: 如果某个个体得到了一个消息，那么他就可能有兴趣把这个消息传播出去（I 态），在传播过程中，他将通过自己的社交媒体账号实时随机地从邻居中选取一人并将消息传给他，如果这个邻居不知道这个消息 (S 态)，则该邻居就会得到该消息并且其状态将可能会变成 I 态，进行下一轮传播。但如果该邻居已经知道了这个消息，那么该邻居就会认为该消息失去了继续传播的价值，并失去传播该消息的兴趣转为 R 态。不同个体间传播消息的概率有差异，不同拓扑结构的网络中传播规律也不同 [12]。整个过程

的示意图如图 2

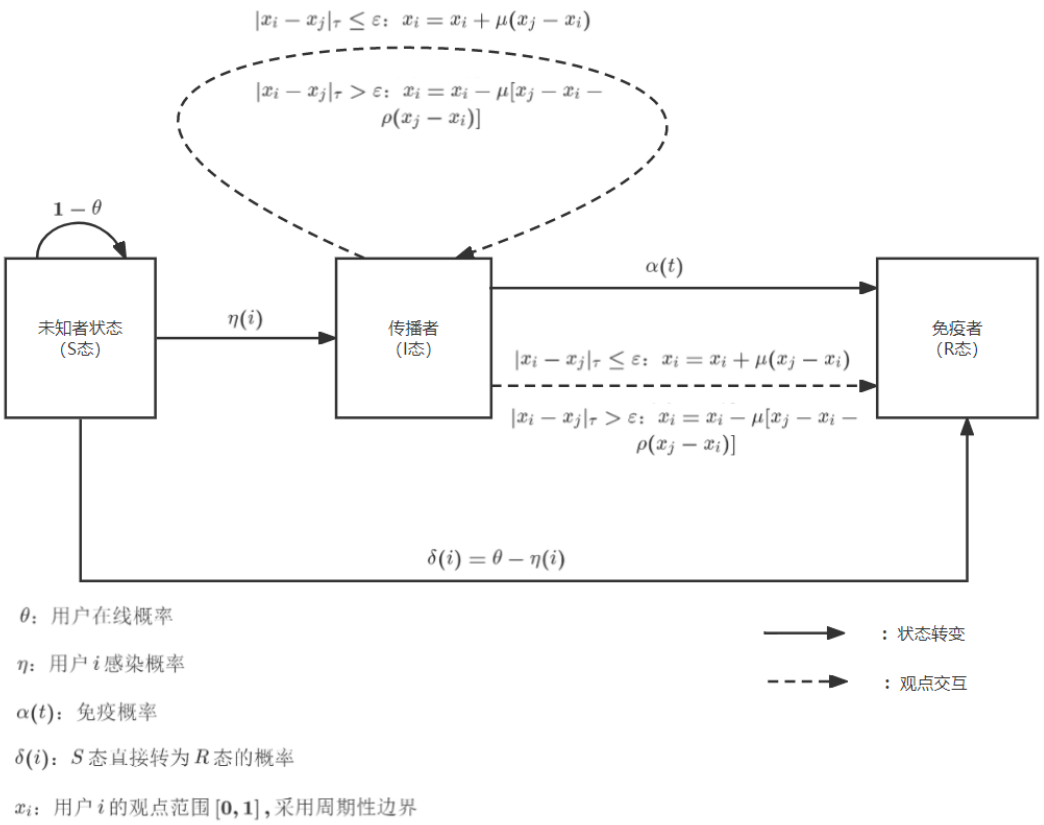


图 **2**　话题在网络上的传播过程示意图

若假设在一个有限网络中所有节点都处于 S 态，在某一时刻突然有一个节点得到了新的消息，它将成为 I 态。按照上面定义的规则，消息将开始传播，直到最终网络中没有 I 态的节点为止，这个状态被称为终态。考虑两个相邻节点 A、B，它们由一条边联系。假定节点 A 知道消息并于时间 t 将消息传给节点 B，则在时刻 $t+1$，节点 B 将会将消息传播给其相邻节点。A 称为 B 的“父”节点，A 与 B 的其他相邻节点所处的地位将会不同。当 A 转而成为 B 传递信息的的相邻节点，按照传播规则，A 将会变为 R 态；而当 B 的其他相邻节点作为 B 传播信息的对象时，B 将根据当时的情形来决定是保留在 I 态还是变为 R 态，即 B 变为 R 态的概率小于 1。如果 A 的度为 k，则 B 的下一传播节点为 A 的概率为 $\frac{1}{k}$，而选择其他相邻结点的概率为 $1-\frac{1}{k}$。设 S 态、I 态、R 态对应的数目分别为 $n_{k,S}$、$n_{k,I}$、$n_{k,R}$，则根据上述分析得到 $n_{k,S}$、$n_{k,I}$、$n_{k,R}$ 的演化方程如式 1.

$$\begin{cases} n_{k,S}(t+1) = n_{k,S}(t) - \sum_{k' \neq k} n_{k',I}(t)(1-\frac{1}{k'})P(k'|k)\frac{n_{k,S}(t)}{N_k} \\ n_{k,R}(t+1) = n_{k,R}(t) + n_{k,I}(t)\left[\frac{1}{k} + (1-\frac{1}{k})\sum_{k' \neq k} P(k'|k)\frac{n_{k',I}(t)+n_{k',R}(t)}{N'_k}\right] \end{cases} \tag{1}$$

式 1中：N_k 为度为 k 的节点数；$n_{k,S}(t)$、$n_{k,I}(t)$、$n_{k,R}(t)$ 分别为 t 时刻度为 k 的节点中处于 S 态、I 态、R 态的节点数。$\frac{n_{k,S}(t)}{N_k}$ 和 $\frac{n_{k',I}(t)+n_{k',R}(t)}{N'_k}$ 源于均匀混合假设 [12]。

$n_{k,I}(t+1)$ 满足守恒条件，如式 2.

$$N_k = n_{k,S}(t+1) + n_{k,I}(t+1) + n_{k,R}(t+1) \tag{2}$$

式 2的含义为度为 k 的所有节点应包含三种状态（S 态、I 态、R 态）。将1改为连续形式，如式3.

$$\begin{cases} \dot{n}_{k,S}(t) = -\sum_{k' \neq k} n_{k',I}(t)(1-\frac{1}{k'})P(k'|k)\frac{n_{k,S}(t)}{N_k} \\ \dot{n}_{k,R}(t) = n_{k,I}(t)\left[\frac{1}{k} + (1-\frac{1}{k})\sum_{k' \neq k} P(k'|k)\frac{n_{k',I}(t)+n_{k',R}(t)}{N'_k}\right] \end{cases} \tag{3}$$

式3中的 $\dot{n}_{k,S}$、$\dot{n}_{k,R}$ 分别代表 $n_{k,S}$、$n_{k,R}$ 对时间 t 的导数。

设 T 为话题从发布到传播结束的整个时间周期，即满足式4.

$$\sum_k n_{k,I}(T) = 0 \tag{4}$$

引入辅助变量 $s_k = \int_0^T n_{k,I}(T)dt$ 以求式3在 $t = T$ 时刻的解。设初始时刻（t=0）网络中有一个个体发布话题，并具有度 k_0，则初始条件如式5.

$$\begin{cases} \mathrm{n}_{k,S} = N_k，\ n_{k,I} = 0，\ n_{k,R} = 0, & \mathrm{k} \neq k_0 \\ \mathrm{n}_{k,S} = N_k - 1，\ n_{k,I} = 1，\ n_{k,R} = 0, & \mathrm{k} = \mathrm{k}_0 \end{cases} \tag{5}$$

从而得到式3在 t=T 时刻的解：

$$\begin{cases} n_{k,s}(T) = N_k exp\left[-\frac{k}{\langle k \rangle N}\sum_{k' \neq k} s_{k'}\left(1-\frac{1}{k'}\right)\right] \\ n_{k,R}(T) = N_k\left\{1 - exp\left[-\frac{k}{\langle k \rangle N}\sum_{k' \neq k} s_{k'}\left(1-\frac{1}{k'}\right)\right]\right\} \\ n_{k,I}(T) = 0 \end{cases} \tag{6}$$

式6中，N 为网络总结点数，而 $\langle k \rangle$ 则为网络中所有节点 v_i 的度为 k_i 的平均值，称为平均度，计算如式7.

$$\langle k \rangle = \frac{1}{N}\sum_{i=1}^{N} k_i \tag{7}$$

可得到在终态时度为 k 处于 S 态和 R 的节点密度，如式8.

$$\begin{cases} \rho_{k,S} = \frac{n_{k,S}(T)}{N_k} = e^{-\alpha k} \\ \rho_{k,R} = \frac{n_{k,R}(T)}{N_k} = 1 - e^{-\alpha k} \end{cases} \tag{8}$$

式8中，$\alpha = \frac{1}{\langle k \rangle N}\sum_{k' \neq k} s_{k'}\left(1-\frac{1}{k'}\right)$ 与网络结构有关。那么，传播过程中 S 态和 R 态的总节点数如式9.

$$\begin{cases} N_S(T) = \sum_k n_{k,S}(T) = -\sum_k N_k e^{-\alpha k} \\ N_R(T) = \sum_k n_{k,R}(T) = N - \sum_k N_k e^{-\alpha k} \end{cases} \tag{9}$$

与之对应的节点密度为：

$$\begin{cases} \rho_S = \frac{N_S(T)}{N} = 1 - \sum_k P(k)e^{-\alpha k} \\ \rho_R = \frac{N_R(T)}{N} = \sum_k P(k)e^{-\alpha k} \end{cases} \tag{10}$$

由式10可知，总密度依赖于网络的度分布 P（k）。

接下来是考虑网络类型。在社交平台上的各类社交账号中，大部分节点的连接数较少，而某些节点却拥有与其他节点的大量链接，特别是名人的账号或者知名媒体账号等诸如此类的社交帐号。表现在度分布上往往具有幂律的形式，即 $P(k) \sim k^{-\gamma}$。为支撑我们的这一结论，我们在微博用户随机选择 20000 多位用户统计他们的粉丝数，结果如图3. 由图可知图3这些账号中的粉丝数的分布近似为幂律形式。同时，我们选择微博上有关河南红码事件的相关发帖账号和评论账号，分析其网络结构，结果如图4所示。由图4可知，在网络往往只有少数账号（即对应节点）拥有较多的链接，即对应的度较大，而大部分节点的链接则较少，对应的度较小。

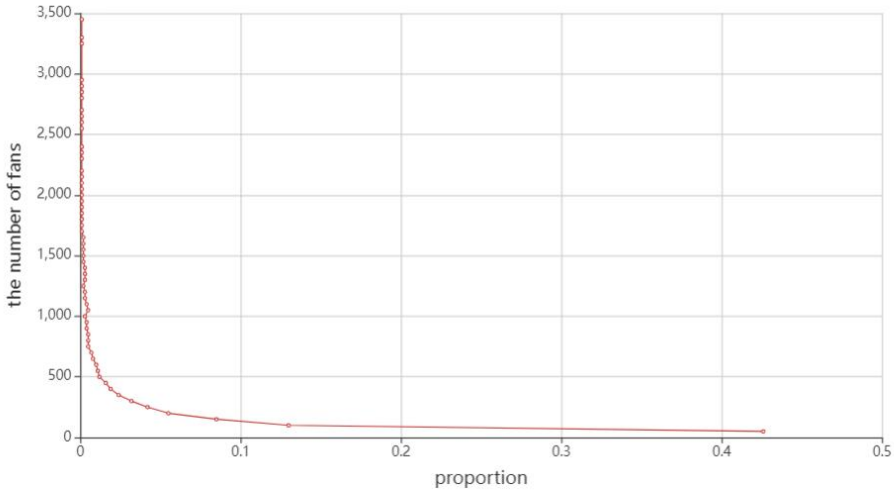


图 **3**　随机抽取微博社交帐号的粉丝数分布统计结果

我们将那些具有大量链接的节点称为“集散节点”，其社交帐号拥有大量关注者。结合 Barabási 和 Albert 在 1999 年提出的一个无标度网络模型 [13]，结合图4的结果，我们将本文所研究网络假设为 BA 无标度网络模型（以下简称 BASFN，即 BA scale-free network）。下面定义网络的两个重要特性:

- 增长特性（Growth）：网络的规模是不断扩大的。例如每天社交媒体上都会有大量新的话题（即视频、帖子等）发布，新的话题往往会引起人们的关注讨论，使得话题的传播越来越广（热度越来越广）。

- 优先连接特性（Prefential attchment）: 新的节点更倾向于与那些具有较高连接度的节点相连接。例如，人们在选择参考文献时往往会选择那些被引次数高的重要文献。

图 **4**　河南红码事件的相关发帖账号和评论账号网络关系

基于上述两个特性得到 BASFN 的构造算法：

(1) 增长。从一个具有 m_0 个节点的网络开始，每次引入一个新的节点，并且连接到 m 个已存在的节点上，m 满足 $m \leq m_0$.

(2) 优先连接。一个新节点与一个已存在的节点 v_i 相连接的概率 Π_i 与节点 v_i 的度 k_i 的关系如式11.

$$\Pi_i = \frac{k_i + 1}{\sum_j (k_j + 1)} \tag{11}$$

在经过 t 步之后，该算法将产生一个有 $N = t + m_0$ 个节点，新增 mt 条边的的网络 [13]。

接下来讨论 BASFN 的度分布。目前对 BASFN 的度分布理论研究主要有三种方法：连续场理论 [13, 14]、速率方程法 [14] 和主方程法 [15, 16, 17, 18]，本文采用主方程法进行分析计算。设网络从 m_0 个孤立节点开始构造，每次引入一个新的节点，并且连接到 m 个已经存在的节点上。定义 $P(k, t_i, t)$ 为在 t_i 时刻加入的节点 v_i 在 t 时刻的度恰好是 k 的概率。考虑到 $\sum_j = 2mt$(因为一条边连接两个节点)，当一个新节点加入到系统中

时，节点 v_i 的度增加 1 的概率为 $m\Pi_i = \frac{k}{2t}$，否则该节点的度保持不变。由此得到递推关系式12.

$$P(k,t_i,t+1) = \frac{k-1}{2t}P(k-1,t_i,t) + \left(1-\frac{k}{2t}\right)P(k,t_i,t) \tag{12}$$

式12的边界条件为 $P(k,t_i,t) = \delta_{km}$, 网络的度分布极限为：

$$P(k) = \lim_{t\to\infty}\left(\frac{1}{t}\sum_{t_i}P(k,t_i,t)\right) \tag{13}$$

式13满足式14. 所示的递推方程。

$$P(k) = \begin{cases} \frac{k-1}{k+2}P(k-1), & k \geq m+1 \\ \frac{2}{m+2}, & k = m \end{cases} \tag{14}$$

从而求得 BASFN 的度分布函数如式15.

$$P(k) = \frac{2m(m+1)}{k(k+1)(k+2)} \propto 2m^2k^{-3} \tag{15}$$

我们将上述过程所建立的传播模型称为 BASFN-SIR 传播模型，将网络视为 BASFN，话题传播动力学机制用 SIR 模型描述。其中式1和式3两种形式动力学方程描述传播过程，而对话题传播终态则可根据式10来预测，结合式15得到话题传播结束时 S 态和 R 态的节点密度，如式16.

$$\begin{cases} \rho_S = \frac{N_S(T)}{N} = 1 - \sum_k \frac{2m(m+1)}{k(k+1)(k+2)}e^{-\alpha k} \\ \rho_R = \frac{N_R(T)}{N} = \sum_k \frac{2m(m+1)}{k(k+1)(k+2)}e^{-\alpha k} \end{cases} \tag{16}$$

5.1.2 SIR-BASFN 模型的模拟分析

为探究影响话题传播的因素，我们设置多组参数进行模拟，结果如后文。

由图5可看出，SIR-BASFN 模型下，在话题传播中 S 态和 I 态的占比逐渐下降，而只有 R 态的占比逐渐上升，这意味着随着时间的推移，网络中没有听说过信息的人逐渐减少，即 S 态的人数越来越少；随着人们对话题熟知，人们会对话题“免疫”，从而失去继续讨论话题、传播话题的兴趣，即 I 态的占比越来越小、R 态的人越来越多，这转而会导致话题传播达到阈值，传播停止或传播速度下降到极慢的状态；同时，我们可以看到图5三个态都是在从零时刻到一个时间点的时间段内，变化率随时间逐渐降低，意味着话题的传播速度随时间的推移逐渐下降。

虽然 SIR-BASFN 模型结果可以大致地描绘话题在社交网络上的传播过程，但由于该模型只考虑了网络拓扑结构和信息传播动力学（SIR 过程），因而该模型有其局限性。比如，该模型没有考虑话题传播过程中用户的观点交换（即用户间的相互影响）会对用户状态（S 态、I 态、R 态）转变的影响，没有考虑不同话题（如话题吸引度）本身对话

表 **2**　改变初始随机感染个节点的模拟参数和结果

总结点数 N	引入新节点时新生成的边数 m	网络初始节点数 m_0	初始随机感染个节点数	对应结果
5000	4	60	300	图5.(a)
5000	4	60	500	图5.(b)
5000	4	60	700	图5.(c)
5000	4	60	1000	图5.(d)

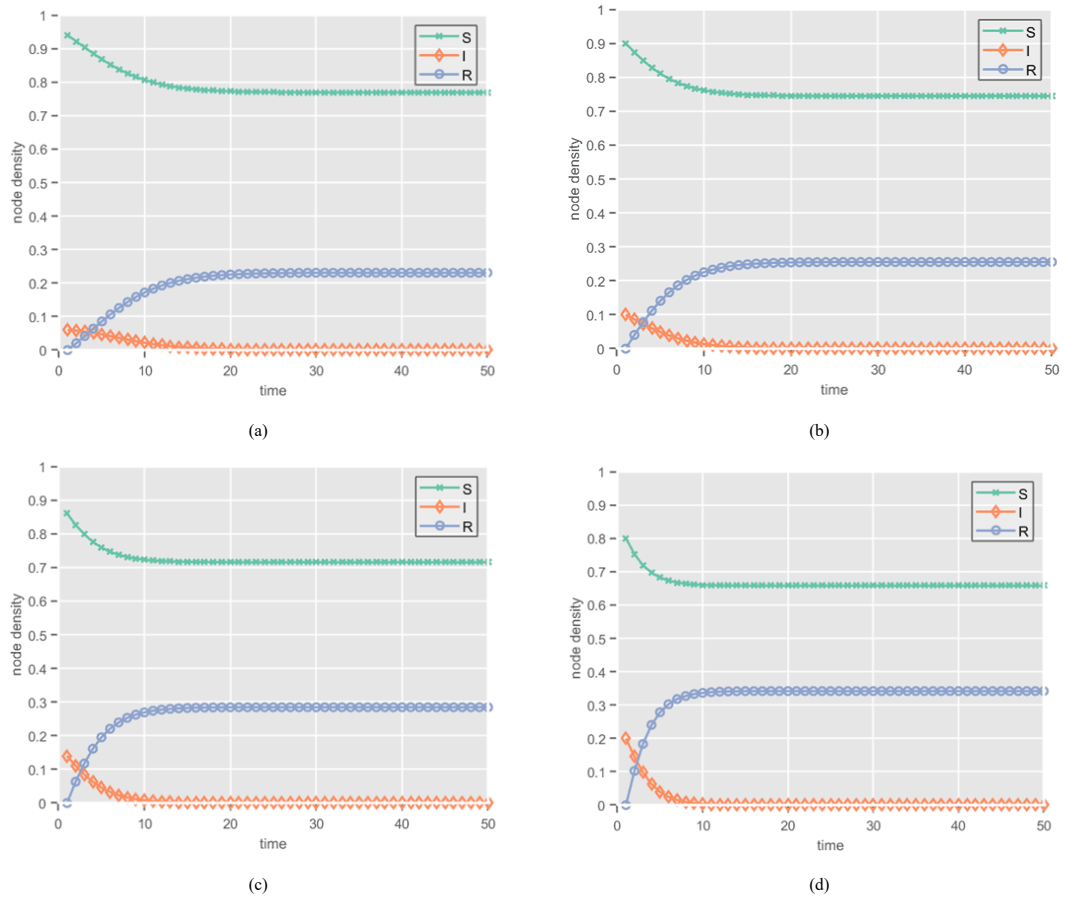


图 **5**　改变初始随机感染个节点的模拟结果

题传播的影响，以及用户心理、平台推荐算法、用户在线活跃度等对对话题传播的影响。接下来，我们会对这些进行简化抽象，构建一个更加完善、考虑因素更多的模型，并分析我们关注的因素对话题传播的影响。

5.1.3 PageRank-SIR-RUCM 模型及其模拟分析

正如图 2中细节所示，我们抽象出并定义了以下变量：用户在线概率 θ；用户感染概率 $\eta(i)$；用户对话题的免疫概率 $\alpha(t)$；用户从 S 态直接转化为 R 态的概率 $\delta(i)$ 以及处于 I 态的用户之间的意见交换过程和 I 态和 R 态之间的意见交换的观点动力学方程（细节请见中立共识和观点极化产生机制分析模型）中的观点值 $x_i(x_i \in [0,1]$，且满足式24所示的周期边界条件函数 $\rho(x)$)、信任阈值 ε（也称容限距离，本文不区分这两个词，指同一概念）和收敛系数 μ（也称趋同系数，本文不区分这两个词，指同一概念）。这里做一下说明，未知者（S 态）由于对信息的不知情，所以我们不考虑 S 态和 I 态、S 态和 R 态之间的意见交换。

首先，我们从用户在社交网络中的心理表现 [19, 20, 21] 出发更新一下网络的形成过程。如图6所示，所有节点都会倾向于与与其观点相近的节点相连接，并遵循出度越大，与新节点相连接的概率越大，依次地增加一定数目的新节点从而构建出网络。

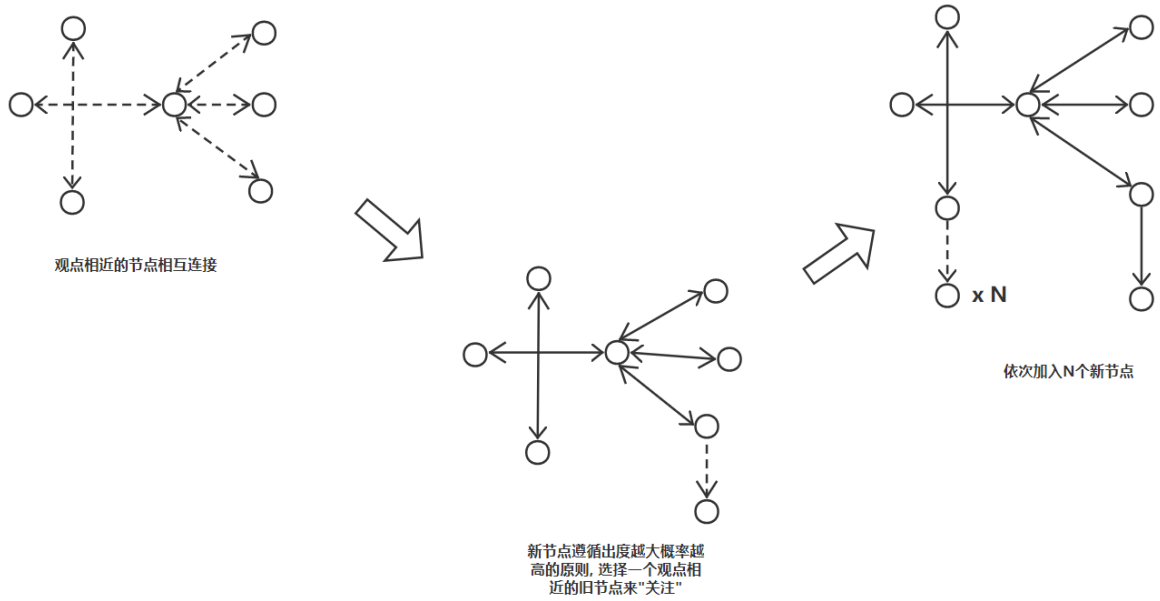


图 **6**　社交网络构建规则

由于在信息传播过程中，随着时间推移，信息的接收者会逐渐对信息失去兴趣，信息的传播能力逐渐降低，接收者的免疫概率会增加，据此定义免疫概率函数如式17.

$$\alpha(t) = \frac{\omega}{1 + e^{\beta - \kappa t}} \qquad 0 < \omega \leq 1, \quad \beta > 0, \quad \kappa > 0 \tag{17}$$

式17中，ω 代表用户对信息的不敏感程度，可理解为传播者的自退化速率；κ 是免疫增长因子；β 是信息传播增长速率相关系数。

接着，我们基于 PageRank 算法来确定网络中不同节点的权威度。PageRank 算法是基于网页链接分析关键字匹配搜索结果进行处理的，它借鉴传统引文分析思想：当网页 A 有一个链接指向网页 B，就认为 B 获得了 A 对它贡献的分值，该值的多少取决于网页 A 本身的重要度，即网页 A 越重要，网页 B 获得的贡献值就越高 [12]。由于网络中网页的相互指向，该分值的计算为一个迭代过程，最终网页根据所得分数进行排序，数学运算过程如下文。

网络是一个有向图，每一个网页视为图的一个顶点，网页间的链接视为图的一条边。邻接矩阵 $\mathbf{B}=(b_{ij})_{N\times N}$(N 为总页面数)，若网页 i 和网页 j 之间有超链接，则 $b_{ij}=1$，否则为 0。记邻接矩阵 **B** 的行和列分别是 $c_j=\sum_{i=1}^{N}b_{ij}$、$r_j=\sum_{j=1}^{N}b_{ij}$，它们分别给出了页面 j 的链入链接数和页面 i 的链出链接数。

假如用户上网时浏览页面并选择下一个页面的过程与过去浏览哪些页面无关，而仅依赖于当前所在的页面，那么这一选择过程可以认为是一个有限状态、离散时间的随机过程，其状态转移规律我们考虑用 Markov 链描述。定义矩阵 $\mathbf{A}=(a_{ij})_{N\times N}$ 如式18.

$$a_{ij}=\frac{1-d}{N}+d\frac{b_{ij}}{r_i},\qquad i,j=1,2,\ldots,N \tag{18}$$

式18中，d 为模型参数，通常取 d=0.85；**A** 为 Markov 链的转移概率矩阵；a_{ij} 为页面 i 转移到页面 j 的概率。

根据 Markov 链的基本性质，对于正则 Markov 链存在平稳分布 $\mathbf{x}=[x_1,\ldots,x_N]^T$，满足式19.

$$\mathbf{A}^T\mathbf{x}=\mathbf{x},\qquad \sum_{i=1}^{N}x_i=1 \tag{19}$$

x 表示在极限状态下（即转移次数趋于无限）各网页被访问的概率分布，Google 将它定义为各网页的 PageRank 值。若能得到 **x**，则它按分量满足式20.

$$x_k=\sum_{i=1}^{N}a_{ik}x_i=(1-d)+d\sum_{i:b_{ik}=1}\frac{x_i}{r_i} \tag{20}$$

网页 i 的 PageRank 值是 x_i，它链出的页面有 r_i 个，于是页面 i 将它的 PageRank 值分成 r_i 份，分别“投票”给它链出的网页。x_k 为网页 k 的 PageRank 值，即网络上所有页面“投票”给网页 k 的最终值。同时，我们也将得到 PageRank 值作为网页的权威度

为了体现受到多个信息的加强效应 [22]，我们定义收到信息的总量为 d(i)，即信息传播者的权威度之和，$d(i)=\sum_{j\in adj(i)}x_j$，其中 adj(i) 是入度邻居的集合。在消息传播过程中，定义未知者 i 接收消息后成为传播者的概率为 $\eta(i)$，表达式如式21.

$$\eta(i)=\theta[1-(1-\lambda)^{hd(i)}] \tag{21}$$

式21中，λ 为信息的固有传播概率，该参数反应信息本身的影响力而与网络结构无关；θ 信息接收者的在线概率；h 为外部社会加强因子，来自社会的关联消息、舆情等均影响 h。很显然，S 态用户 i 直接转化成 R 态的的状态转换概率为 $\delta(i) = \theta - \eta(i)$。

到目前为止，我们已经叙述完了 PageRank-SIR-RUCM 模型主要相关理论，该模型考虑了社交网络中用户因由个人兴趣爱好而与他人产生关联所定义的网络构建过程（即用户心理所致的网络构建过程）、用户之间的相互影响、话题吸引度、平台推荐算法、外部影响因素和用户活跃度。接下来，我们基于 MATLAB 设计模拟程序进行仿真。首先，我们要先检验 PageRank-SIR-RUCM 模型的可靠性。PageRank-SIR-RUCM 模型生成的网络结构如图7所示，图中节点越大的节点代表其度越高，与图4相比较，我们发现两者具有极大的相似性，都是存在少数度很大的节点，而大部分节点则度相对较小。这在一定程度上反应了该模型的结果较为符合实际情况。

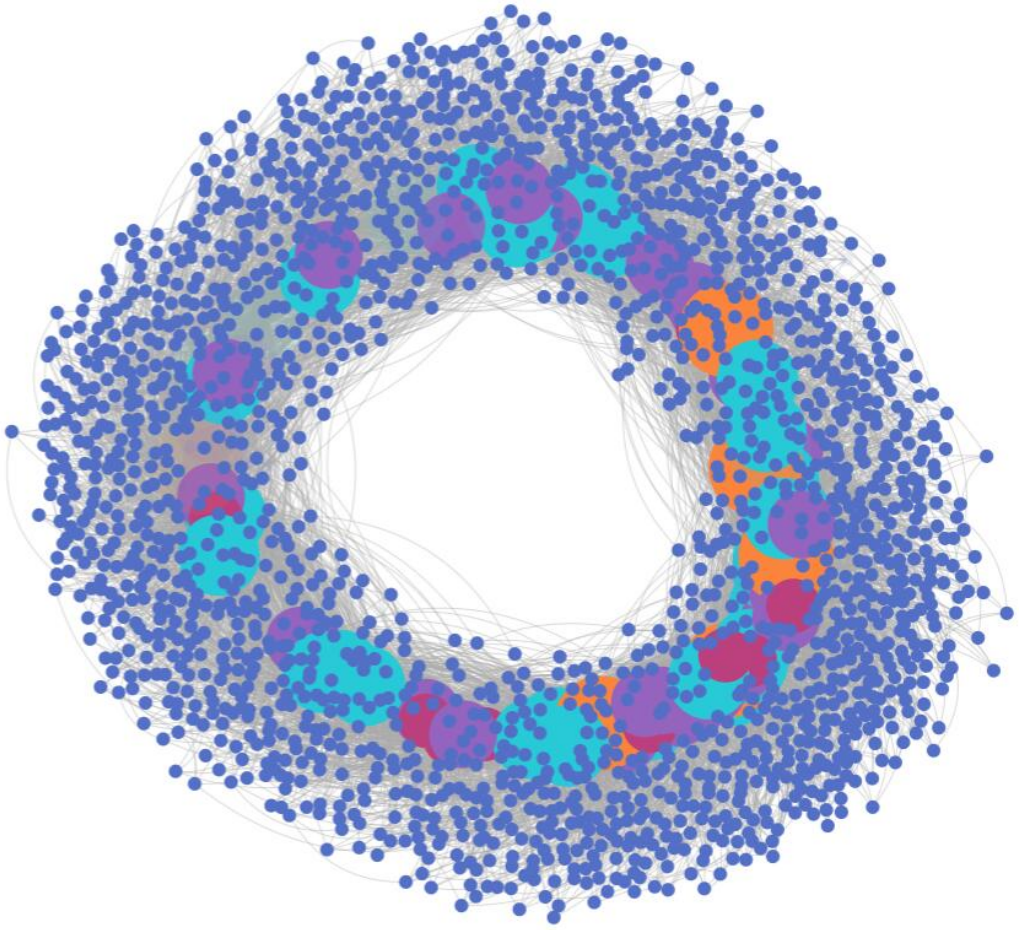
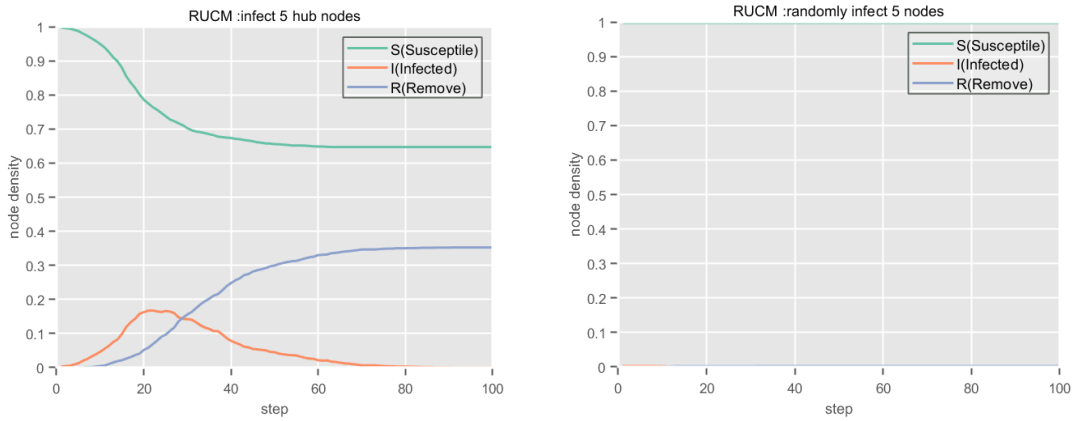


图 7　**PageRank-SIR-RUCM** 模型生成的网络结构

接下来是对影响话题传播的因素进行分析。我们考虑到社交网络中存在一些有着大量粉丝的账号，但这些账号占少数，而大多数账号则关注者较少（由图3和图4可以证明），有着大量粉丝的账号我们称为枢纽节点，这些节点的度较大，而其他节点则称为



(a) (b)

图 8 **PageRank-SIR-RUCM** 模型下选择枢纽节点和普通节点作为初始感染节点的模拟对照结果。初始感染节点为 **5** 个初始节点（**a**），初始感染节点为 **5** 个普通节点（**b**）

普通节点。我们通过选择初始节点为 5 个枢纽节点和 5 个普通节点的情形进行模拟对照分析，来看一下这些具有较大的度的节点作为初始感染节点对话题传播的影响，结果如图8。显然枢纽节点可以较快地将话题传播到较广的范围，而普通节点则几乎没有将话题传播起来。由此可见，初始感染节点的度对于话题传播的影响显著。接着，我们综合考虑模型中涉及的参数的影响。我们采用控制变量的方式对各个参数对话题传播过程的影响进行分析，结果如图17。图17含有 8 个子图，对应的操纵和模拟结果分别是：

改变用户活跃度即用户在线概率 θ（a）；改变用户对信息的敏感程度 ω(属于用户心理)（b）；改变信息传播增长速率相关系数 β（信息本身特性）(c)；改变免疫增长因子 κ（用户心理）(d)；改变信息固有传播概率 λ(e)；改变信任阈值 ε（用户心理）(f)；改变网络平均度 $\langle k\rangle$(网络结构)（g）；改变外部社会加强因子 h（环境影响因素如舆情、政府干预等）（h）.

由图17结果可知，上述参数中改变信息固有传播概率 λ 会使得 S 态节点密度下降速度变快，即可以很大程度上促进话题传播，而其他参数的影响则不是很显著。

综上所述，对于话题传播的影响最大的两个因素是初始感染节点的度和话题固有性质——话题传播概率。

5.1.4 两类话题的数据分析

我们针对微博，使用爬虫爬取大量话题数据，最终确认“河南村镇银行多位储户又被赋红码”为中立共识话题，而“二舅治好了我的精神内耗”为观点极化的话题。

首先分析“河南村镇银行多位储户又被赋红码”。“河南村镇银行多位储户又被赋红码”该话题的宏观数据如图9所示，该话题的宏观数据属于双峰结构，话题从发起到传播结束经历了两个峰值，历时近九天。为探究该微博用户话题的观点，我们需要对微博话题的评论的情感态度进行分析。本文采使用的中文情感分析的架构使用了一个叫 TNet

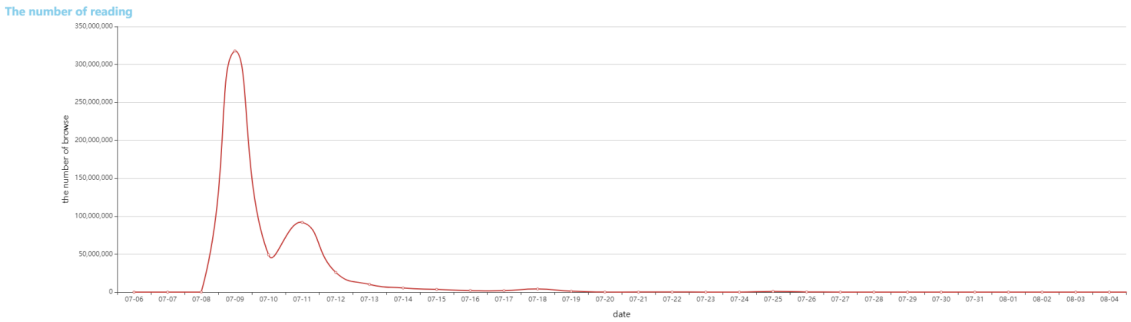
（Transformation Networks）[23] 的目标特定转换网络，TNet 首先将短文信息编码为单词嵌入，并使用 LSTM 保存单词上下文的语境。为了将目标信息集成到单词表示法中，TNet 使用了一个新的特定目标转换（Target-Specific Transformation）组件用于产生目标特定词的意思。与传统基于注意力的方法相反，注意力方法应用相同的目标表征来确定上下文单句的注意力分数，TST 首先根据个体条件生成目标的不同表征上下文词，然后它合并每个上下文的单词及其量身定制的目标表示法获得变换后的单词表示。如图10所示，底层是 BiLSTM，其将输入转换为语境化的词表示（即 BiLSTM 的隐藏状态），中间部分就是我们 TNet 网络的核心，由 L 个上下文保存转换（Context-Preserving Transformation）层组成。CPT 层包括通过新的目标特定变换将目标信息转换为单词表示（TST）的组件。CPT 还包含一种上下文保护机制，类似于身份映射和主要连接，允许保存上下文信息和使用深度网络学习更加抽象的词性特征。网络的顶层大部分是具有方位感知的卷积层，其首先编码词与句子中的方位相关性，然后提取特征信息用于分类。目前国内的公司腾讯有基于 TNet 的语义分析开源 API（https://cloud.tencent.com/document/product/271/35497）。我们基于腾讯的开源语义分析 API 对“河南村镇银行多位储户又被赋红码”话题下的用户评论进行分析，最终以 sentiment_score 来定量描述用户的态度。sentiment_score 的取值区间为 [0,1], 越靠近 0 代表用户的态度越负面，越靠近 1 代表用户的态度越正向。“河南村镇银行多位储户又被赋红码”话题下的所有用户评论的 sentiment_score 统计结果如图11所示。由图11可知，该话题在 2022-7-08 到 2022-7-11 的这几天内，态度为负面的人数的占比显然远远多于态度为正向的人数。因此，我们可以认为该话题为中立共识的类型。

按照与分析“河南村镇银行多位储户又被赋红码”话题相同的方法和流程，我们得到“二舅治好了我的精神内耗”话题的宏观数据如图12所示。该话题的宏观数据为单峰结构，话题发起到传播接近尾声历时近 8 天。同样地，采用 sentiment_score 来分析该话题下人们的观点，统计结果如图13所示。由图13可知，态度为负向的的人数与态度为正向的人数比列相近，故我们认为该话题属于观点极化的类型。

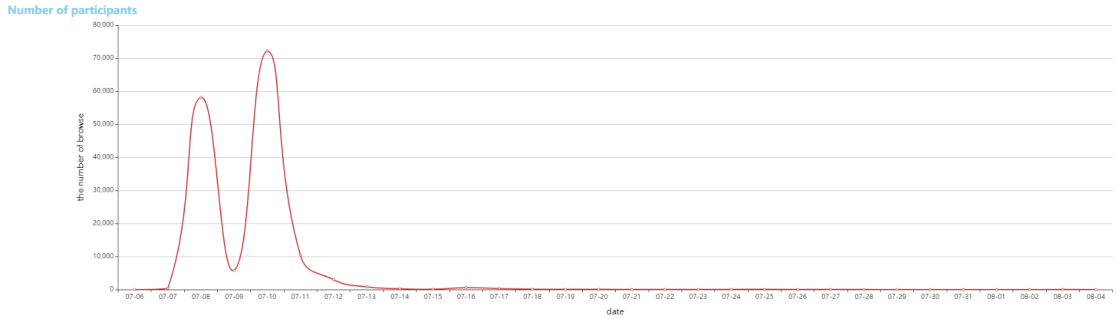
5.2 问题二分析与求解

5.2.1 中立共识和观点极化产生机制分析

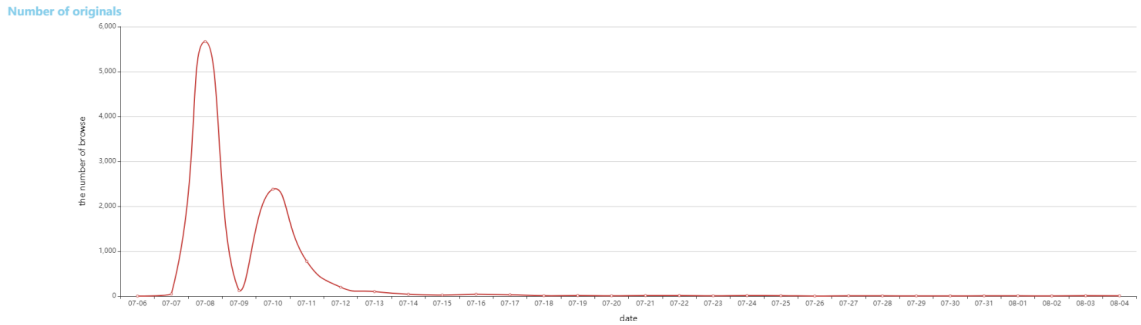
人们倾向于认同符合他们的认知体系的信息而忽略不赞同的信息，网络内容的广泛可用性促进了志同道合者的聚集，而辩论则往往会使得群体观点的极化 [24, 25, 26, 27, 28, 29]。观点同质化（即趋于一致）往往是网络中话题传播驱动力，两极分化和同质化可能是认知偏爱和社会影响结合的结果 [30, 22, 31]。人们在认知偏爱和社会影响的基础上塑造自己的观点，这两种力量的结合产生观点极化和中立共识 [32]。为量化研究这两个现象的形成机制，我们在有界置信重线模型（the Rewire with Bounded Confidence Model, 以下简称 RBCM）[33]、有界置信模型（the classical Bounded Confidence Model, 以下简



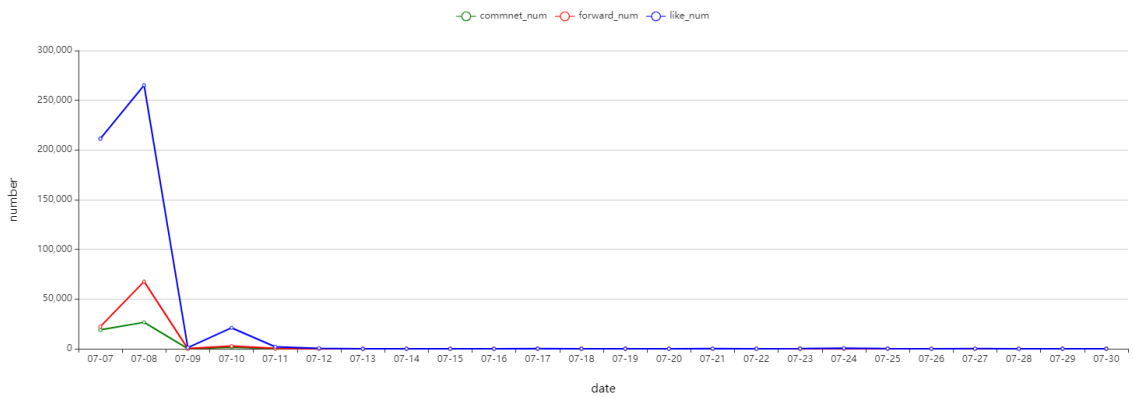
(a) 话题阅读量变化趋势



(b) 话题讨论变化趋势



(c) 话题原创始人数变化趋势



(d) 话题评论、转发、点赞人数变化趋势

图 **9**　“河南村镇银行多位储户又被赋红码”话题宏观数据总览

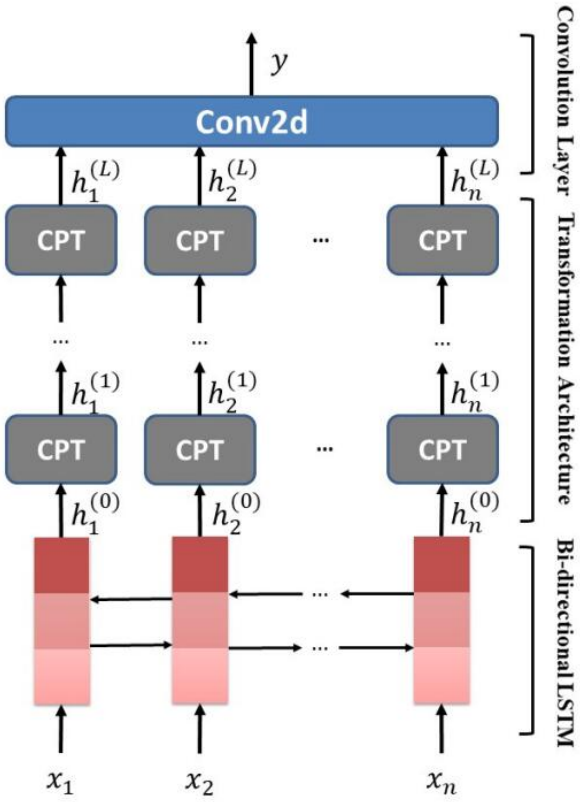
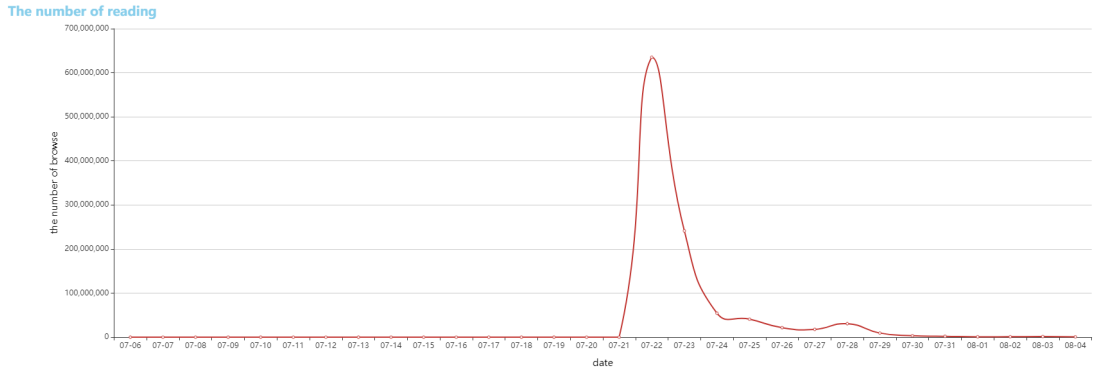
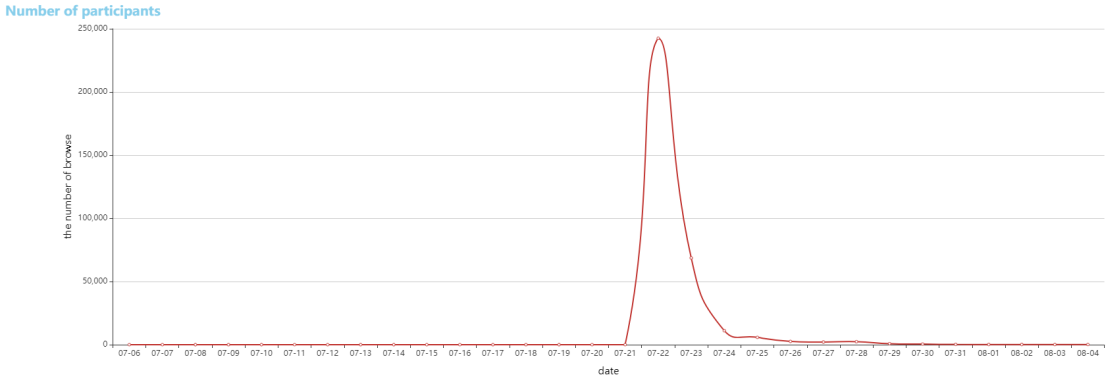


图 **10**　**TNet** 的算法架构 **[23]**

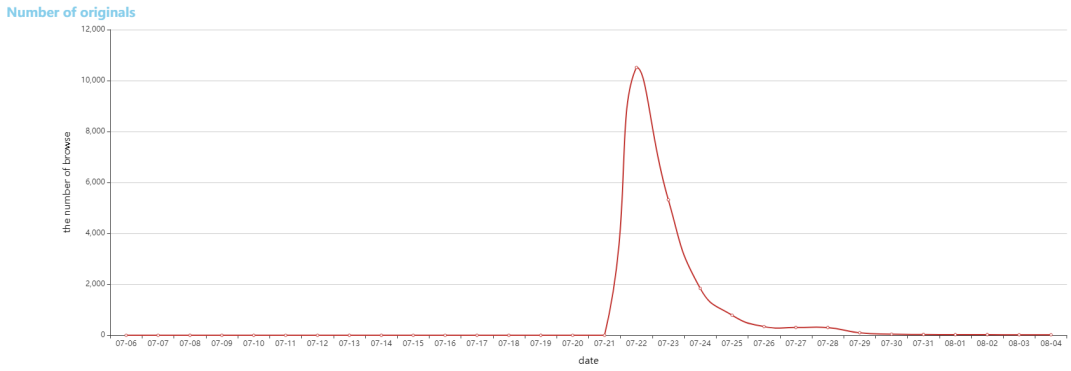
图 **11**　“河南村镇银行多位储户又被赋红码”话题下的所有用户评论的 **sentiment_score** 统计结果



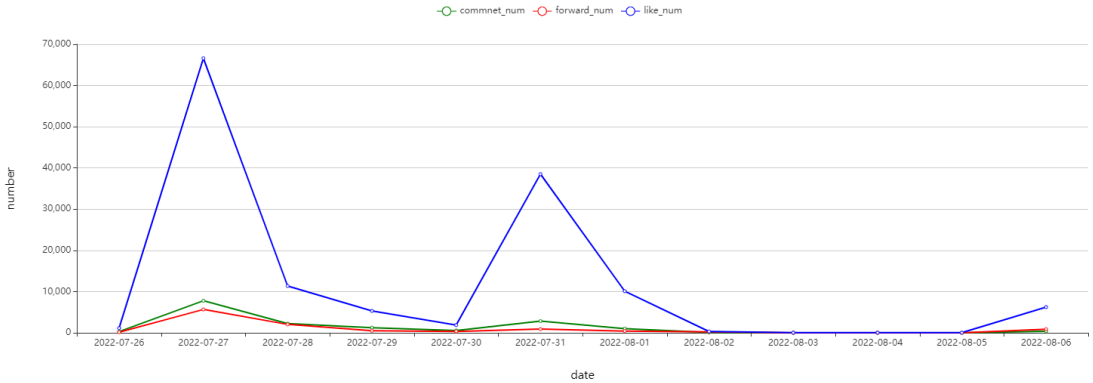
(a) 话题阅读量变化趋势



(b) 话题讨论变化趋势



(c) 话题原创始人数变化趋势



(d) 话题评论、转发、点赞人数变化趋势

图 **12** “二舅治好了我的精神内耗”话题宏观数据总览

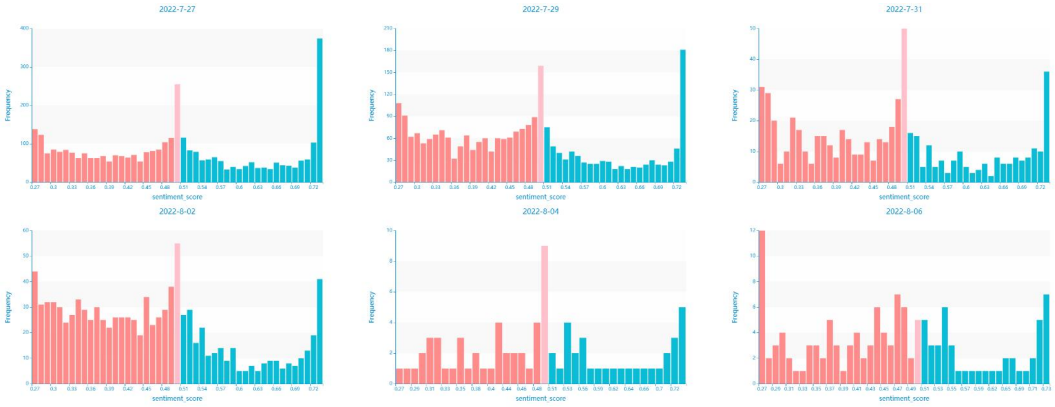


图 **13** “二舅治好了我的精神内耗”话题下的所有用户评论的 **sentiment_score** 统计结果

称 BCM）[34, 35, 36]、无界置信模型（the Unbounded Confidence Model, 以下简称 UCM）[33]、无界置信重线模型（the Rewiring Unbounded Confidence Model, 以下简称 RUCM）[33] 的基础上展开研究。其中，RBCM 下网络中不一致的链接会被打破，直至达到收敛；UCM 允许不一致的用户进行交互和引入负向更新规则，在 UCM 的基础上增加重新布线规则可得 RUCM（细节请见后文）；对于 BCM，则假设意见区间是连续的。

首先，我们定义有界置信度附带条件：

- 如果信息或意见足够接近代理状态 (agent state，此处代理理解为社交媒体用户)，则与之交互。

对于 BCM，将 N 个代理构成的集合安排在复杂网络 G 上，每个代理持有观点 $x_i, i \in \{1, \ldots, N\}$，且均匀分布于 [0,1] 之间。当且仅当两个代理在 G 上有连接和他们所持有的观点足够接近时（即 iif $j \in N_G(i)$ and $|x_i - x_j| < \varepsilon$, for $\varepsilon \in [0, 1]$）条件成立时，两个代理按照式22规则改变他们的观点，否则他们不发生交互。

$$\begin{cases} x_i = x_i + \mu(x_j - x_i) \\ x_j = x_j + \mu(x_i - x_j) \end{cases} \tag{22}$$

对于 RBCM，需要考虑两个阶段。

(1) 在第一个阶段，我们执行重布线步骤，每个代理 i 随机选择邻居 j 进行交互，但如果两个代理对应的观点的距离超过容限距离 ε, 他们的连接会被打破，同时，i 随机选择代理 k（$k \in 1, \ldots, N/(N_G(i) \cup i)$）进行重新布线。具体来说，我们引入新的距离 $|.|_\tau : [0, 1] \times [0, 1] \to [0, 0.5]$，定义如式23.

$$|x_i - x_j|_\tau = |x_i - x_j - \rho(x_i - x_j)| \tag{23}$$

式23中，$i,j \in \{1,\ldots,N\}$，调节参数 ρ 满足周期性边界条件（PBC），定义如式24.

$$\rho(x) = \begin{cases} -1, & if \quad x \in [-1,-0.5) \\ 0, & if \quad x \in [-0.5,0.5] \\ 1, & if \quad x \in (0.5.1] \end{cases} \tag{24}$$

随机重新布线的条件变为：$|x_i - x_j|_\tau \geq \varepsilon$，for $\varepsilon \in [0,0.5]$. 注意这里在条件 $\forall x,y \in \{1,\ldots,N\}$ 得到 $|x-y|_\tau \in [0,0.5]$ 后，限制关注范围为 $\varepsilon \in [0,0.5]$。当所有连接的观点距离低于容限 ε 时，第一阶段结束。

(2) 在第二个阶段，我们在重新布线的网络上执行 BCM。BCM 只允许意见距离小于容限 ε 的两个代理相互作用，由于重布线网络中的所有偶对（有联系的两个代理）都是一致的，因此，所有随机选择的偶对将根据式22所定义的规则发生相互作用并重新调整意见，其中 μ 只能在（0，0.5）内取 [33]。

对于 UCM，允许每一对随机选择的邻居（i，j）发生相互作用。具体地讲，若两个代理的意见一致，即 $|x_i - x_j|_\tau < \varepsilon$, 在前面的模型中按照式22来调整 x_i 和 x_j。但当他们的意见不一致时，即若 $|x_i - x_j|_\tau \geq \varepsilon$，则定义式25来表示新的更新规则。

$$\begin{cases} x_i = x_i - \mu[x_j - x_i - \rho(x_j - x_i)] \\ x_j = x_j - \mu[x_i - x_j - \rho(x_i - x_j)] \end{cases} \tag{25}$$

式25中，μ 在（0，0.5）中取，而 $\rho(.)$ 的定义见式24。调节参数 $\rho(.)$ 通过使意见保持在区间 [0,1] 内来保证 PBC 的适用性。

对于 RUCM，依然允许随机选择的代理对（i，j）发生相互作用，同时，也允许意见不一致的代理对重新布线。特别地，若 $|x_i - x_j|_\tau < \varepsilon$，则根据式22来更新 x_i、x_j；若 $|x_i - x_j|_\tau \geq \varepsilon$，则根据式25更新。节点 i 和节点 j 之间的连接被打破，同时随机选择代理 k（$k \in \{1,\ldots,N\}/N_G(i) \cup \{i,j\}$）与 i 建立新连接。

以上就是四个模型的观点交换更新规则和代理之间的连接的重新布线规则。为探究哪一种模型能更好的适用于我们在问题一中建立的 BASFN 网络模型，我们将调节参数进行模拟。

首先通过问题一中建立 BASFN 的算法构建一个节点数 N=2000 的网络, 在此网络中对 BCM,RBCM,UCM,RUCM，采取不同的参数组合（$(\varepsilon,\mu) \in [0,0.5] \times [0,0.5]$），采用蒙特卡罗算法模拟观点在 $x \in [0,1]$ 之间的概率密度函数 PDF。注意到系统总能在 10^7 步之前达到最终状态, 对观点频率分布采用局部极大值法估算 PDF 峰的数量 (具体地说，我们将区间 [0，1] 做 20 等分，然后分别计算各区间的观点所占的比例。当峰的高度达到 0.02，且与其它峰之间的距离超过 0.1 时，将此峰计入总的峰值数)，重复 5 次取平均得到如图14所示的结果，图中的色阶表示峰值的数量（由深蓝到浅黄代表峰数由 $1 \rightarrow 8$）。根据图14的结果，有大量的参数能够使得网络中代理（即网络用户）的观点呈极化和趋

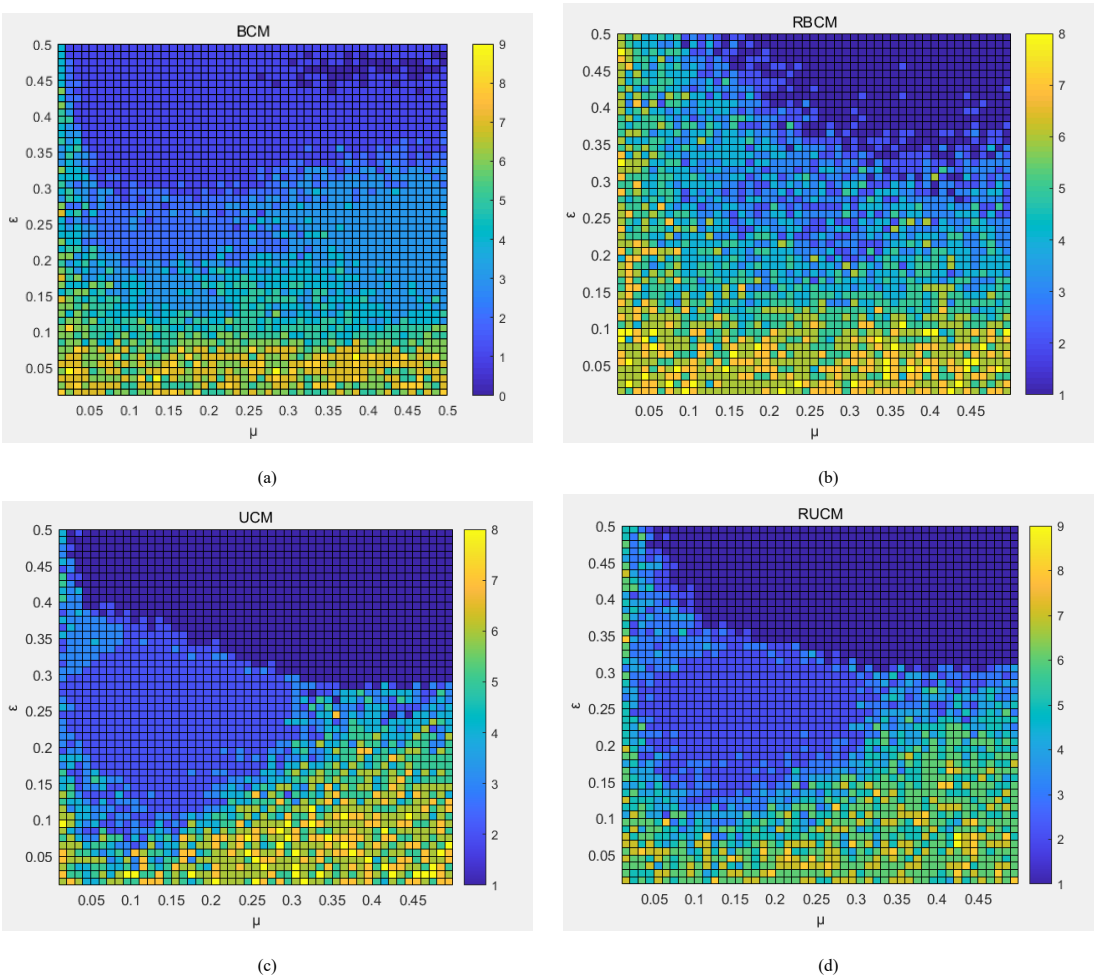


图 **14** $BCM, RBCM, UCM, RUCM$ 在不同的参数组合（$(\varepsilon, \mu) \in [0, 0.5] \times [0, 0.5]$）下的蒙特卡洛模拟峰数结果

于一致的两种状态。我们选取两组参数（$\varepsilon = 0.25, \mu = 0.15$) 和（$\varepsilon = 0.4, \mu = 0.45$）分别得到观点极化和中立共识的现象，如图15所示。

接下来，我们需要分析两个参数 ε 和 μ 的实际意义，以完全解释中立共识和观点计划的产生机制。参数 ε 反映了用户心理，即用户偏好等，而 μ 则反应了不同用户间的趋同性。此外，我们发现 PageRank-SIR-RUCM 模型也可以得到观点极化和中立共识现象，模拟参数如表3, 结果如图16.

5.2.2 尖叫效应形成机制

根据尖叫效应的定义 [1]，可以推知当存在尖叫效应时，在网络信息传播中由尖叫效应引起的话题传播过程具有传播速度快、传播范围广的特点。结合问题一中图8和图17的分析结果，我们知道当初始感染节点的度较大，且话题固有传播概率较大时话题的传播速度显著提升，这意味引起尖叫效应的主要原就是初始感染节点的度较大，且话题固有传播概率较大。

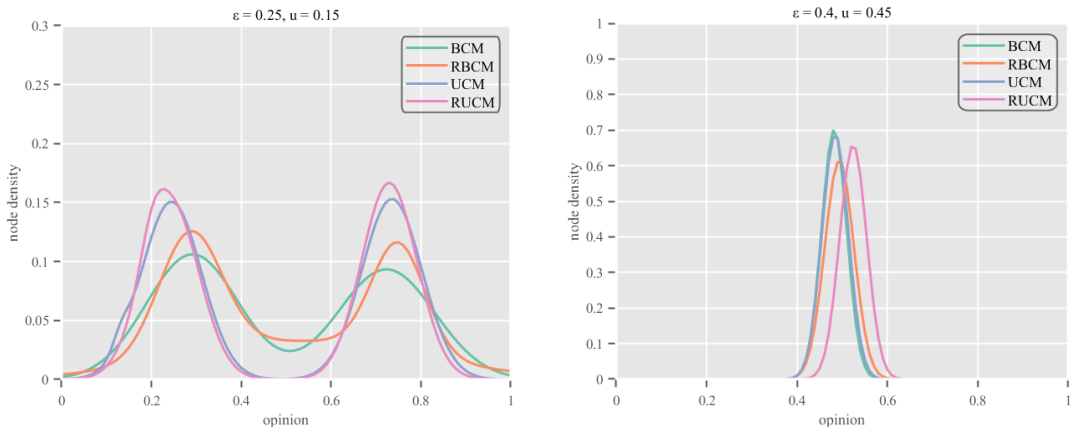
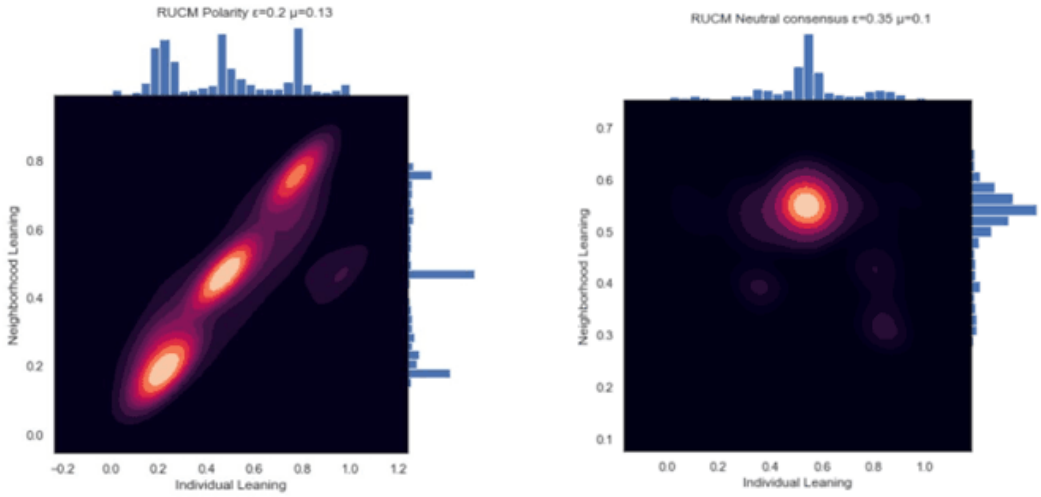


图 **15** $BCM, RBCM, UCM, RUCM$ 在两组的参数组合 $\varepsilon = 0.25, \mu = 0.15$**)** 和（$\varepsilon = 0.4, \mu = 0.45$）分别得到观点极化（**a**）和中立共识的现象（**b**）

表 **3** **PageRank-SIR-RUCM** 模型模拟观点极化和中立共识模拟参数

参数	设置
总节点数 N	2000
平均度 k	5
初始节点数 m_0	100
用户在线概率 θ	0.5
外部社会增强因子 h	2
网络用户对消息的不敏感度 ω	0.1
β	5
κ	1
λ	0.8

话题在网络中的传播速度与范围可以简单的用 S 态节点随时间的演化来刻画，为展示一个尖叫效应案列，我们在 PageRank-SIR-RUCM 模型下分别选定度比较大的 1 个节点和 5 个节点作为初始感染节点，同时设置话题固有传播概率 λ 从大到小进行模拟，结果如图18。由图18可以直观地看出，当度大的节点的个数越多，话题固有传播概率 λ 越大，S 下降的越快，且下降的越低。由于 S 态代表未知者，所以 S 态的节点密度的斜率可以代表话题的传播速度，而其节点密度则可以反向推出接收到话题的人，从而可以得



(a) (b)

图 16 **PageRank-SIR-RUCM** 模型下调节参数得到用户及其邻居的观点核密度图。观点极化（**a**），中立共识点（**b**）

到话题的传播范围。因此，当初始感染者的度越大且初始感染者的数目越多，话题固有传播概率 λ 越大，则发生尖叫效应越明显。

5.2.3 回声室效应形成机制

社交媒体会限制人们接触不同的观点，并倾向于形成志同道合的用户群体，形成并强化一种共享的叙事，这就是回音室，表现为社交媒体用户喜欢与他们信念一致的信息，并加入围绕一个共同叙事而形成的群体 [37, 38, 39, 40, 41]。我们可以将回音室广义地定义为这样一个环境：在这样的环境中用户关于某个话题的观点、政治倾向或信念会因为与具有相似倾向和态度的同行或信息员的反复互动而得到加强，同时会在社交网络中形成相对较小的社区，在这个相对较小的社区里，远离常识的信念通常会得到加强。形成回音室首先要形成一个封闭的用户群，在这里，人们在特定的观点上保持一致。这样的现象与物理学中的自发对称性破缺 (Spontaneous symmetry breaking)[42] 的概念相似。文献 [43] 利用自发对称性破缺来解释社交网络中观点极化的机制，文献 [44] 采用自发对称性破缺来解释在线回声室效应，我们将结合这两篇文献来阐明社交网络中的回声室效应的形成机制数学模型。

首先，我们先简要的介绍一下震荡模型 [44] 的基本公式，以为后文做铺垫。对于一个有向图 G(V,E), 节点 $i, j \in V$，用来代表网络中 N 个用户的结构。若有向连接（$i \to j$）$\in E$ 的权重定为 w_{ij}, 相邻矩阵 $\boldsymbol{\mathcal{A}} = [\mathcal{A}_{ij}]_{1 \le i,j \le n}$ 定义如式26.

$$\mathcal{A}_{ij} := \begin{cases} w_{ij} & (i \to j) \in E \\ 0 & (i \to j) \notin E \end{cases} \tag{26}$$

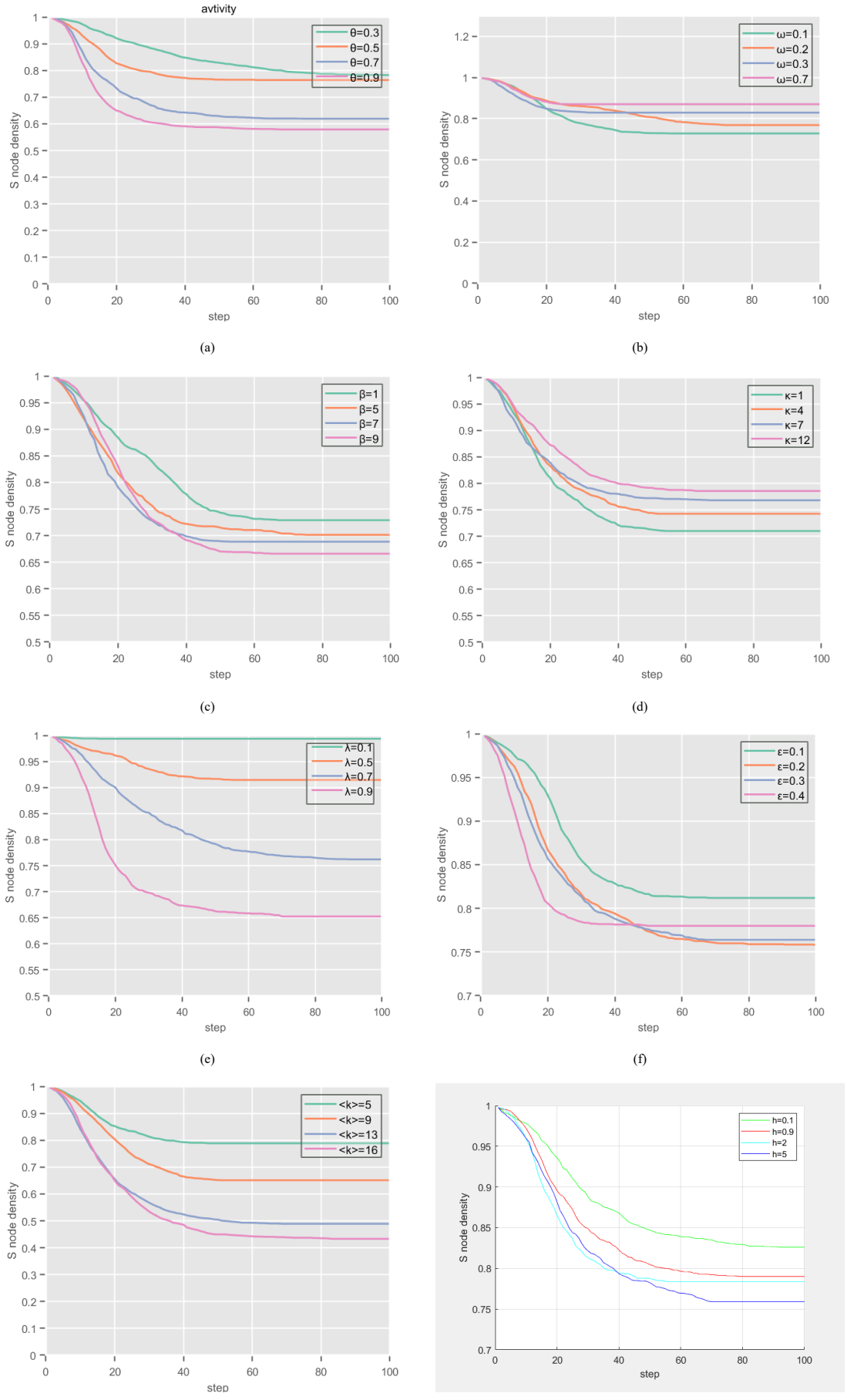


图 **17**　**PageRank-SIR-RUCM** 模型下采用控制变量法进行模拟，**S** 态节点密度对照结果。

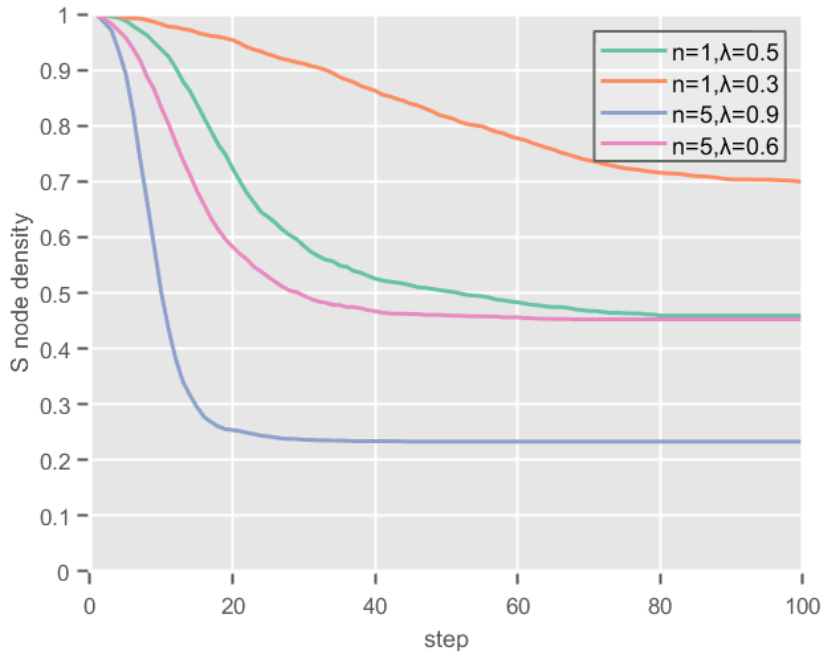


图 **18** **PageRank-SIR-RUCM** 模型下调节度大的初始感染节点个数 **n** 与话题固有传播概率 λ 的模拟结果

同时，给出节点（加权）出度 $d_i := \sum_{j \in \partial j} w_{ij}$，度矩阵的定义如式27.

$$\boldsymbol{\mathcal{D}} := diag(d_1, \ldots, d_n) \tag{27}$$

式27中，∂i 代表从节点 i 开始的出链路的相邻节点的集合。接下来，网络结构的拉普拉斯矩阵由式28来定义。

$$\boldsymbol{\mathcal{L}} := \boldsymbol{\mathcal{D}} - \boldsymbol{\mathcal{A}} \tag{28}$$

这里假设 $\boldsymbol{\mathcal{L}}$ 的所有特征值都是实数。设 t 时刻用户的态矢量为：

$$\boldsymbol{x}(t) := {}^t(x_1(t), \ldots, x_2(t)) \tag{29}$$

式29中的 $x_i(t)(i = 1, \ldots, N)$ 代表节点 i 在时间 t 时的用户状态。则网络的波动方程写为式30

$$\frac{d^2}{dt^2}\boldsymbol{x}(t) = -\boldsymbol{\mathcal{L}}\boldsymbol{x}(t) \tag{30}$$

这里除了需要求解波动方程式30的解 $\boldsymbol{x}(t)$ 之外，还需要考虑网络结构对用户动态的影响，即我们需要思考网络结构和用户动态之间的因果关系。为达到这个目的，我们需要建立一个关于时间的一阶微分方程（以下简称基本方程）[45, 46]。基本方程式一个一阶微分方程，并且它的解也是式30的解。这里介绍一种简单易实现的方法，通过使用一个正半定矩阵 $\sqrt{\boldsymbol{\mathcal{L}}}$（即 $\boldsymbol{\mathcal{L}}$ 的平方根），我们引入如式31所示的方程。

$$\pm i\frac{d}{dt}\boldsymbol{x}^{\pm}(t) = \sqrt{\boldsymbol{\mathcal{L}}}\boldsymbol{x}^{\pm}(t) \tag{31}$$

两个基本方程（式31）可以用一个 2 态矢量表示为一个表达式，如式32

$$i\frac{d}{dt}\hat{\boldsymbol{x}}(t)=\left(\sqrt{\boldsymbol{\mathcal{L}}}\otimes\begin{bmatrix}+1 & 0\\ 0 & -1\end{bmatrix}\right)\hat{\boldsymbol{x}}(t) \tag{32}$$

对于 $\boldsymbol{x}^{\pm}(t)={}^{t}(\boldsymbol{x}_1^{\pm}(t),\ldots,\boldsymbol{x}_n^{\pm}(t))$，二维态矢量 $\hat{\boldsymbol{x}}(t)$ 的定义如式33.

$$\hat{\boldsymbol{x}}(t):=\boldsymbol{x}^{+}(t)\otimes\begin{pmatrix}1\\0\end{pmatrix}+\boldsymbol{x}^{-}(t)\otimes\begin{pmatrix}0\\1\end{pmatrix} \tag{33}$$

这里将式32称为 Boson-Sype 基本方程，简称 BSFE。即使拉普拉斯矩阵 $\boldsymbol{\mathcal{L}}$ 是稀疏的，其平方根矩阵 $\sqrt{\boldsymbol{\mathcal{L}}}$ 一般也是一个完整的图，如图19所示 [44]。但对于实际的社交网络，假

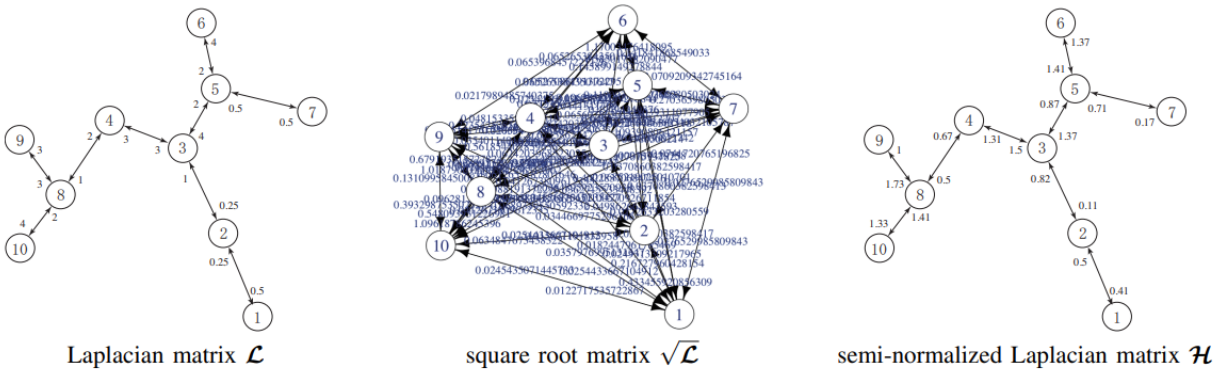


图 **19** 拉普拉斯矩阵 $\boldsymbol{\mathcal{L}}$、平方根矩阵 $\sqrt{\boldsymbol{\mathcal{L}}}$、半标准化拉普拉斯矩阵 $\boldsymbol{\mathcal{H}}$ 对应的网络结构样例 **[44]**

设所有用户都是直接连接的情况是不合理的，基本方程中出现的矩阵必须能够完全反应网络的链路结构（网络节点中是否存在链路）。因此，BSFE（式32）的解在非完全图结构的网络中不可能存在。为解决该问题，我们引入其他基本方程。引入一个新的矩阵半标准化拉普拉斯矩阵 $\boldsymbol{\mathcal{H}}$，如式34.

$$\boldsymbol{\mathcal{H}}:=\sqrt{\boldsymbol{\mathcal{D}}^{-1}}\boldsymbol{\mathcal{L}}=\sqrt{\boldsymbol{\mathcal{D}}}-\sqrt{\boldsymbol{\mathcal{D}}^{-1}}\boldsymbol{\mathcal{A}} \tag{34}$$

式34中，$\sqrt{\boldsymbol{\mathcal{D}}}:=diag(\sqrt{d_1},\ldots,\sqrt{d_n})$。而标准化拉普拉斯矩阵如式35.

$$\boldsymbol{\mathcal{N}}:=\sqrt{\boldsymbol{\mathcal{D}}^{-1}}\boldsymbol{\mathcal{L}}\sqrt{\boldsymbol{\mathcal{D}}^{-1}}=\boldsymbol{\mathcal{I}}-\sqrt{\boldsymbol{\mathcal{D}}^{-1}}\boldsymbol{\mathcal{A}}\sqrt{\boldsymbol{\mathcal{D}}^{-1}} \tag{35}$$

35中，$\boldsymbol{\mathcal{I}}$ 是 $N\times N$ 的单位矩阵。半标准化拉普拉斯矩阵 $\boldsymbol{\mathcal{H}}$ 也是一个有向图的拉普拉斯矩阵，当忽连接是否存在时，它与 $\boldsymbol{\mathcal{L}}$ 有相同的略链路结构，如图19所示。利用半标准化拉普拉斯矩阵 $\boldsymbol{\mathcal{H}}$ 可以得到新的用户动力学方程，如式36.

$$i\frac{d}{dt}\hat{\boldsymbol{x}}(t)=\left(\boldsymbol{\mathcal{H}}\otimes\hat{\boldsymbol{a}}+\sqrt{\boldsymbol{\mathcal{D}}}\otimes\hat{\boldsymbol{b}}\right)\hat{\boldsymbol{x}}(t) \tag{36}$$

式36中 $\hat{\boldsymbol{a}}$ 和 $\hat{\boldsymbol{b}}$ 均为 2×2 的矩阵，定义如式37，

$$\hat{\boldsymbol{a}}=\frac{1}{2}\begin{bmatrix}+1 & +1\\ -1 & -1\end{bmatrix},\qquad \hat{\boldsymbol{b}}=\frac{1}{2}\begin{bmatrix}+1 & -1\\ +1 & -1\end{bmatrix} \tag{37}$$

$\hat{\boldsymbol{x}}(t)$ 依然是一个二维态矢量。式31的解 $\boldsymbol{x}(t)$ 可以由 $\hat{\boldsymbol{x}}(t)$ 按式38得到。

$$\boldsymbol{x}(t)=(\boldsymbol{\mathcal{I}}\otimes(1,1))\,\hat{\boldsymbol{x}}(t) \tag{38}$$

式36和相对论量子力学中的狄拉克方程类似，它的特点是 $\hat{\boldsymbol{a}}$ 和 $\hat{\boldsymbol{b}}$ 满足式39所示的反对易关系。

$$\{\hat{\boldsymbol{a}},\hat{\boldsymbol{b}}\}:=\hat{\boldsymbol{a}}\hat{\boldsymbol{b}}+\hat{\boldsymbol{b}}\hat{\boldsymbol{a}}=\hat{\boldsymbol{e}},\qquad \hat{\boldsymbol{a}}^2=\hat{\boldsymbol{b}}^2=\hat{\boldsymbol{o}} \tag{39}$$

式39中 $\hat{\boldsymbol{e}}$ 表示一个 2×2 的单位矩阵，$\hat{\boldsymbol{o}}$ 表示一个零矩阵。因而我们又将36称为 Fermion-Type 基本方程，简称 FTFE。

假设一个相对较小的子网对应于一个封闭的社区，从其底层在线社交网络中分离出来，隔离的子网成为一个完整的图，如图20。与隔离前的网络相比，被隔离的子网可以在不同的平衡点上震荡，从而实现偏置状态。除 FTFE 的解之外，BTFE 还可能存在一个新的解，因为隔离子网实际上是一个完整的图。我们将基于 FTFE（式36）展示一

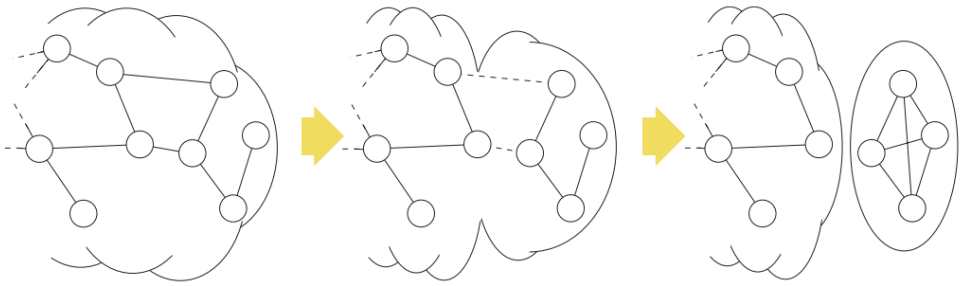


图 **20**　网络结构变化形成孤立的集群 **[43]**

个最简单的例子。首先考虑这样一个情形：所有隔离的子网链路的权值都等同于 w。在此情况下，用户之间的关系强度的连接的权重已增加到极限，并且由于孤立的社区讨论完全被激活而达到饱和。此时，若隔离的社区中的用户数为 n，则所有节点的度相同，$d=(n-1)w$。同时，在相应的拉普拉斯矩阵中，除 0 之外的所有特征值相同，记相同的特征值为 $\lambda=w^2=nw$。此时，由于度矩阵为 $\boldsymbol{\mathcal{D}}=d\boldsymbol{\mathcal{I}}$，则矩阵 $\boldsymbol{\mathcal{H}}$ 和 $\sqrt{\boldsymbol{\mathcal{D}}}$ 可以被同时对角化。因此，FTFE（式36）进行变换，使 $\boldsymbol{\mathcal{H}}$ 和 $\sqrt{\boldsymbol{\mathcal{D}}}$ 都被对角化，从而 FTFE 可以表示为 2×2 的块对角化形式。所以拉普拉斯矩阵除 0 之外的所有特征值的变换方程都

是相同的，可以写成式40.

$$\begin{aligned} i\frac{d}{dt}\boldsymbol{\psi}(t) &= \left(\frac{w^2}{2\sqrt{d}}\begin{bmatrix}+1 & +1\\ -1 & -1\end{bmatrix} + \frac{\sqrt{d}}{2}\begin{bmatrix}+1 & -1\\ +1 & -1\end{bmatrix}\right)\boldsymbol{\psi}(t) \\ &= \begin{bmatrix}+\frac{w^2+d}{2\sqrt{d}} & +\frac{w^2-d}{2\sqrt{d}}\\ -\frac{w^2-d}{2\sqrt{d}} & -\frac{w^2+d}{2\sqrt{d}}\end{bmatrix}\boldsymbol{\psi}(t) \end{aligned} \tag{40}$$

若式40的二维向量解记为式41.

$$\boldsymbol{\psi}(t) = \begin{pmatrix}\psi^+(t)\\ \psi^-(t)\end{pmatrix} \tag{41}$$

设式41的解为

$$\psi^{\pm}(t) := exp(\mp i\theta^{\pm}(t)) \tag{42}$$

从而由式40、41、42可得：

$$\frac{d}{dt}\theta^{\pm}(t) = \frac{w^2+d}{2\sqrt{d}} + \frac{w^2-d}{2\sqrt{d}}exp(\pm i(\theta^+(t)+\theta^-(t))) \tag{43}$$

由于 $\theta^{\pm}(t)$ 是一般复数形式，将 $\theta^{\pm}(t) = Re[\theta^{\pm}(t)] + iIm[\theta^{\pm}(t)]$ 带入43得到式44.

$$\begin{cases} \frac{d}{dt}Re\theta^{\pm}(t) = \frac{w^2+d}{2\sqrt{d}} + C^{\pm}(t)sin\left(-(Re[\theta^{\mp}(t)] - \frac{\pi}{2}) - Re[\theta^{\pm}(t)]\right) \\ \\ \frac{d}{dt}Im\theta^{\pm}(t) = \pm C^{\pm}(t)sin\left(-(Re[\theta^{\mp}(t)] - \frac{\pi}{2}) - Re[\theta^{\pm}(t)]\right) \end{cases} \tag{44}$$

式44中，$C^{\pm}(t) := \frac{w^2-d}{2\sqrt{d}}exp(\mp(Im[\theta^+(t)] + Im[\theta^-(t)]))$。根据这个结果（式44）可以预测其以下性质：

值得注意的是，因为 $C^{\pm}(t) > 0$，$\theta^{\pm}(t)$ 实部的时间演化（式44）具有和 Kuramoto model[47] 类似的结构。与 Kuramoto model[47] 不同的是，$C^{\pm}(t) > 0$ 不是一个常数。如果 $C^{\pm}(t) > 0$ 足够大，并且 Kuramoto model[47] 发生相位同步，实现 $Re[\theta^+(t)] + Re[\theta^-(t)] = \frac{\pi}{2}$ 状态，与此同时，式44虚部变为 $\frac{d}{dt}Im[\theta^{\pm}(t)] = \pm C^{\pm}(t)$。那么会发生 $\theta^+(t)$ 随时间的增加而增加，$\theta^-(t)$ 随时间的增加而减小。但根据式42，两种变化情况都会使 $\psi^{\pm}$ 的幅值增大。这个结果意味着被隔离的社区的用户活动被激活，上述过程描述了网络中的回音室效应的发生过程。

虽然上述模型可以解释回音室的产生机制，但比较抽象，不利于直观理解回音室形成后对应的现象。文献 [41] 提出一种描述社交媒体上的回音室的量化分析模型，我们将基于文献 [41] 构建一个简单直白的刻画网络中的回音室的量化分析模型。首先可将回音室广义地定义为这样一种环境，在这种环境中，个体关于某个话题的观点、政治倾向或信念会因为与具有相似倾向和态度的同辈或信息员反复互动而得到加强 [41]。因而要判断一个用户是否处于回音室中，关键在于要分析他和他的邻居的情感倾向，若一个个体和他的邻居（与某个个体能够发生联系的其他用户）的情感倾向一致，那么他们的就

会有较大概率形成回音室并陷入回音室当中。设个体 i 对一个对特定的话题 T_i 的个人倾向可以通过不同的方式推断，如通过个体产生的内容 [48]。这里，我们将倾向定义为一段内容对某一特定主题所表达的态度。考虑个体 i 产生数量为 a_i 的内容 c_i，所有的 c_i 形成一个话题集合 C_i，$C_i = \{c_1, \ldots, c_{a_i}\}$。这里，每一个内容的倾向 c_i 都会通过问题一模型中使用的 TNet 算法通过分析话题内容来赋一个数值 ($c_i \in [0, 1]$)。那么一个个体的倾向就可以由该个体产生的话题的话题倾向的平均值来量化，如式45.

$$T_i = \frac{\sum_{j=1}^{a_i} c_j}{a_i} \tag{45}$$

网络的拓扑结构可以显示出回音室，具有回音室效应的网络中，个体被具有相似倾向的其他个体包围，用网络术语来说，具有给定倾向 T_i 的节点更有可能与倾向为 T_i 的节点连接 [40]。这个概念可以被量化定义，对于一个个体 i，它的邻居的平均倾向可以定义为如式46.

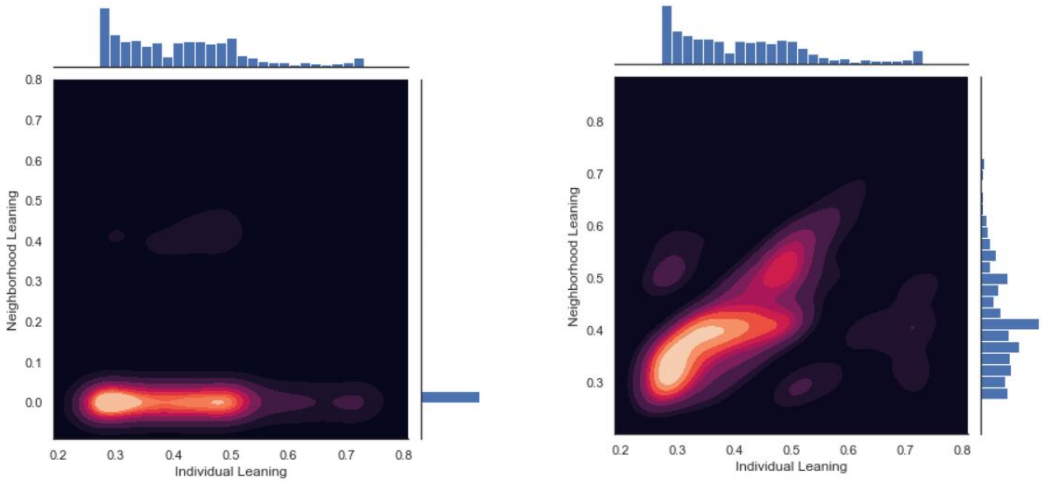
$$T_i^N = \frac{1}{k_i^{\rightarrow}} \sum_j A_{ij} T_j \tag{46}$$

式46中，A_{ij} 是网络的邻接矩阵，若节点 i 与节点 j 之间有一条链接则 A_{ij}=1，否则为 0（此处只要网络中的两个用户可以交换观点就认为存在这样的链接），而 $k_i^{\rightarrow}$ 则是节点 i 的出度，$k_i^{\rightarrow} = \sum_j A_{ij}$。接下来将网络中所有节点及其邻居的倾向量化后做成相图 (T, T^N)。若网络中节点和它的邻居的倾向相近，则相图的特征为在两坐标轴之间的四十五度对角线上的密度较高且相对集中表示网络中的回声室效应越强。我们针对微博话题河南村镇银行多位储户又被赋红码，爬取数据量为 6574，以及唐山烧烤店打人，数据量是 9843，将评论内容传入腾讯有基于 TNet 的语义分析开源 API（https://cloud.tencent.com/document/product/271/35497）获取情感倾向分数（范围 0 到 1），将该情感分数赋于 c_i，得到两个话题下的社区的相图 (T, T^N) 如图21所示，可以看到话题唐山烧烤店打人对应的社区的回声室效应更明显。

许志源等通过 2016 年美国总统大选分析了“过滤气泡”现象与启示，并指出信息茧房（Information cocoons）现象、回音室（Echohamber）效应与过滤气泡（Filter bubble）这三个名词是对同一现象赋予的不同名字，三者所针对的是同一现象 [49]。为节省时间和简化问题，我们将回音室效应的影响因素分析与信息茧房影响因素分析也同样视为同一个问题，留在下文进行分析和求解。

5.2.4 信息茧房形成机制

本文将沿用“中立共识和观点计划产生机制”模型中的观点动力学的分析视角建立信息茧房的形成机制模型。首先我们抽象出信息建房的三个主要建模因素：价值认同、信息同质性、社区影响。价值认同是个体对信息所蕴含的价值的认同而形成的价值观念。作为社会的人，个体会根据自身的个性或后天的经验形成相应的价值观。如果个体



(a) (b)

图 **21** 两个社区网络中节点及其邻居的倾向相图 (T, T^N)。（**a**）话题河南村镇银行多位储户又被赋红码社区的相图 (T, T^N)，其回音室效应较弱，（**b**）话题唐山烧烤店打人对应社区的相图 (T, T^N)，回音室效应较强

接收到的信息与自己的价值观一致，就会产生愉悦感。一般个体在处理信息时，总会根据自己的需求、价值观来选择不同的信息，学习与信息固有价值体系相同的信息，并主动拒绝不匹配的信息 [37, 38, 39, 40, 50, 48, 51]。基于其自己的价值特征，它呈现出一定的信息选择模式。基于这种认知心理，个体被划分为不同的圈子，在圈子内表现出趋同心理甚至从众心理，而在圈子外表现出一定程度的排外心理。随着时间的推移和算法技术的日益强化，这种信息选模式容易形成圈内同质化、圈外异质性的社会现象，从而产生信息茧房效应，由外而内由浅向深不断发展 [52]。现代社会产生的算法工具基于语义片段构建用户画像，能够精准推送与用户价值观念高度重合的信息。一旦用户被各种同质化的信息包围，往往会导致价值认同、信息同质化，从而形成信息茧房。意见动力学认为意见的演变是人与环境相互作用的结果，社会网络是影响个体构筑信息环境的主要因素。现实世界中的社会网络是由圈子构成，其中社会关系强，价值观相似，相互影响大，个体之间的价值认同也强；而圈子之间的社会关系相对较弱，价值观不同，相互影响小，个体之间的价值认同较弱。一个由价值观相近的人构成的社区是相对封闭、稳定的。两个社区中的个体长期处于具有相同价值观的群体中，不接触具有不同价值观的群体时，容易产生群体极化，导致信息茧房的出现。由此可见，社区结构对信息茧房的形成至关重要。以个人为社交网络的节点，以人际关系为链接而形成复杂网络，就是意见互动所依赖的互动网络。我们将通过社区内外信任阈值的差异来表达不同社区之间的价值差异，从而反映社区内部的价值认同和社区内部对同质信息的接受程度。

首先，从结构上定义社区网络，社区呈现出无标度特征，社区之间联系紧密，而在社区之间的联系相对稀疏；社区代表同意领域的群体，不同社区具有异质性，在初始阶

段，意见值设置在不同的区间内。社区内部关系紧密，价值认同强，它们共享相似的价值，一般接受同质信息，因此信任阈值 δ 较大。而社区间的社会关系弱、价值认同不同，因此社区间的信任阈值 δ 较小。这里，社区中的信任阈值 δ 表达了社区中的个体的一种价值认同。社区 1 内部的信任阈值记为 δ_1，社区 2 内部的信任阈值为 δ_3，两个社区间的信任阈值为 δ_2，则应有 $\delta_2 < \delta_1$,$\delta_2 < \delta_3$。下面定义相互作用规则：

社区中并不是所有的个体都可以进行互动，只有有联系的个体才能进行互动，这代表了一种现有的社会网络关系。在任意时刻，随机选择两个个体 i 和 j 进行交互。$o_i(t) \in [0,1]$ 代表个体 i 在时间 t 时的意见值，$d_{ij} = |o_i(t) - o_j(t)|$ 代表个体 i 和 j 之间的意见距离。当 $d_{ij} = |o_i(t) - o_j(t)| < \delta$（$\delta$ 是能够发生观点互动的阈值），个体 i 和 j 将会互动。在每一轮中，社区 1 中的个体被随机选择，与与其相连的邻居进行互动。如果邻居在社区 1 中，则交互信任阈值 δ_1 比较大，交互过程如式47.

$$\begin{cases} d_{ij} = |o_i(t) - o_j(t)| < \delta_1, & \delta_2 < \delta_1 \\ o_i(t+1) = o_i(t) + \zeta(o_j(t) - o_i(t)) \\ o_j(t+1) = o_j(t) + \zeta(o_j(t) - o_i(t)) \end{cases} \tag{47}$$

若邻居不在社区 1 中，则相互信任的阈值相对较小，交互规则如式48.

$$\begin{cases} d_{ij} = |o_i(t) - o_j(t)| < \delta_2, & \delta_2 < \delta_1 \\ o_i(t+1) = o_i(t) + \zeta(o_j(t) - o_i(t)) \\ o_j(t+1) = o_j(t) + \zeta(o_j(t) - o_i(t)) \end{cases} \tag{48}$$

对于社区 2 亦是如此。

探索信息茧房的实验算法设计如下：

（1）按照 SIR-BASFN 模型中生成 BASFN 的算法构造社区互动网络，社区 1 度为 k_1，总结点数为 N_1，社区 2 度为 k_1，总结点数为 N_1，在社区之间的链接数为 m。

(2) 为每个个体赋一个初始观点值。社区 1 内的 N_1 个个体的观点均匀分布在 [a,b] 之间，社区 2 内的 N_2 个个体的观点均匀分布在 [c,d] 之间 ($0 \le a < b \le b$,$-1 \le c < d \le 0$)。个体 i 在时刻 t 的观点值为 $o_i(t)$。

(3) 设置信息茧房模型的信任阈值。社区 1 的信任阈值为 δ_1，社区 2 的信任阈值为 δ_3，在社区之间的信任阈值为 δ_2，收敛系数为 ζ（也称为趋同系数）。仿真中的具体参数设置如表4。

(4) 根据设定的交互规则，对信息茧房模型进行 500 次迭代的实验仿真。

我们将该模型称为基于观点动力学与 BASFN 的茧房形成机制模型，通过 MATLAB 编程进行仿真，在几组参数下得到如图22所示的结果，在一些参数下两个社区形成两个收敛峰，而在一些参数下则形成多个峰，形成两个峰代表两个社区的连接较少或没有，即代表在这两个社区中形成了信息茧房。根据仿真结果可以得出规律：随着社区内的信

表 **4**　信息茧房形成机制模型模拟参数

编号	主要参数	值
1	个体数 (N)	1000
2	意见空间 (O)	$O \sim U(-1,1)$
3	模拟步数 (t)	2×10^4
4	社区 1 节点的度 (k_1)	23
5	社区 2 节点的度 (k_2)	23
6	社区间的链接数 (m)	[0,1000]
7	社区 1 的信任阈值 (δ_1)	[0,1]
8	社区间的信任阈值 (δ_2)	[0,1]
9	社区 2 的信任阈值 (δ_3)	[0,1]
10	收敛系数 (ζ)	0.3
11	主流观念 (O_c)	[-1,1]

任阈值 δ 的升高，最终两个社区的观点将会收敛到较窄的观念值区间，即导致茧房的产生。

接下来，分析话题吸引度、用户活跃度、用户心理、不同用户间的影响、平台推荐算法等因素对形成信息茧房的影响。我们将帖子作为有向网络中的节点，每个帖子带有一个观点，范围在 [-1,1]。首先建立一个规模较小的网络，将所有观点近似的帖子连接起来（信任阈值为 ϵ_1）,$A \to B$ 连接意为帖子 A 存在一个超链接指向 B，类似社交媒体平台推荐算法生成的“猜你还喜欢”这类的链接推荐。接着每天加入一定数量的帖子并与网络相连接，新增的节点将在信任阈值 ϵ_1 下优先与入度高的节点连接，发生连接的概率为 $P(k_i) = \frac{k_i}{\sum_i k_i}$。随着帖子的增多，我们发现网络将会逐渐呈现出无标度网络的特点，如图23所示，网络的度分布逐渐呈现幂律分布。

我们每天会向网络中添加一些节点，接着令用户在网络节点之间游走，以此来模拟用户在网上冲浪的行为。用户的观点范围是 [-1,1], 假设用户有一个初始观点，根据用户初始观点我们可以确定用户在由帖子构成的网络中的初始位置：信任阈值 ϵ_2 内，入度越

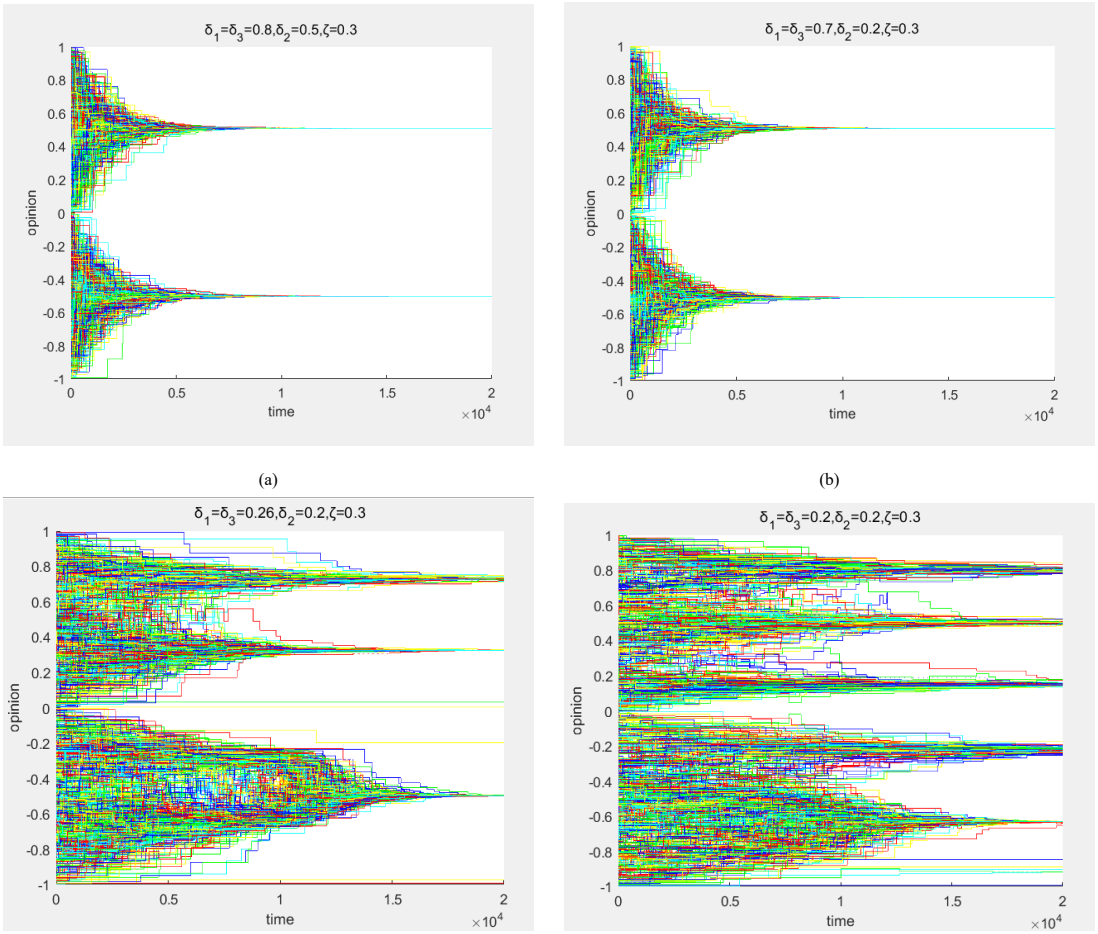


图 **22**　不同的参数组合（$(\delta,\zeta)\in[0,1]\times 0.3$）下的两个社区观点值的收敛结果

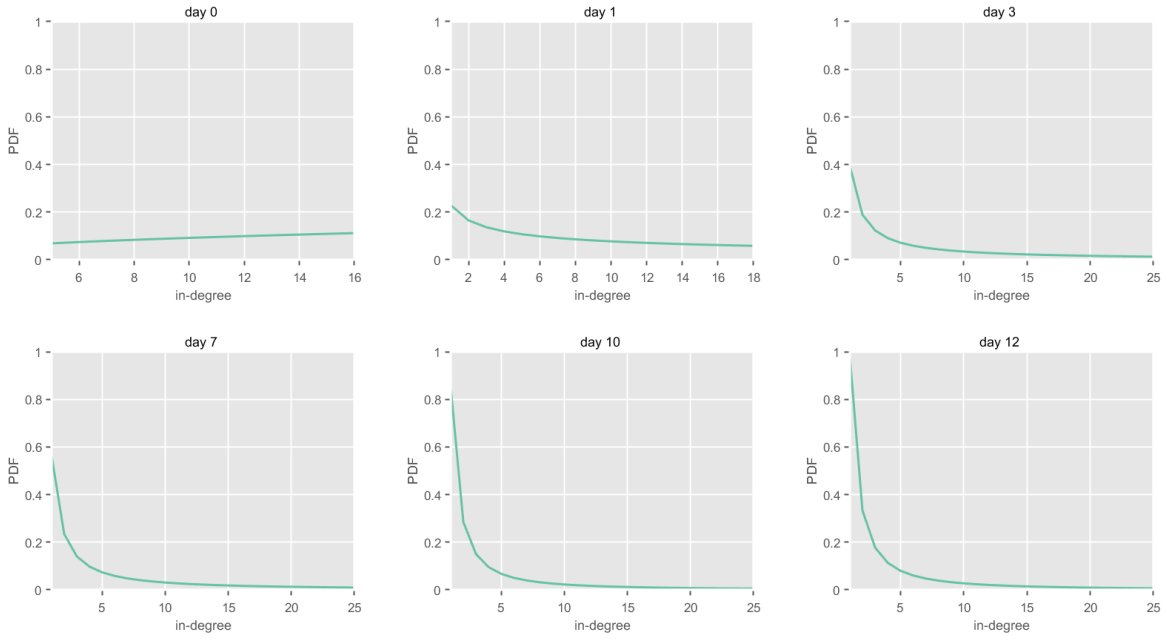


图 **23**　网络演化过程中度分布的演化

大的节点被选为初始节点的概率越高。这里我们将信息茧房抽象地定义为：用户在一定的随机游走步数内没有走到超出信任阈值 ϵ_2 的节点，则视用户陷入了信息茧房（再次提醒，ϵ_1 是帖子之间形成超链接的信任阈值，ϵ_2 是用户对帖子和其他用户的信任阈值）。我们选择使用 TOPSIS（Technique for Order Preference by Similarity to an Ideal Solution）[53] 来确定用户在节点间游走的转移概率。引入 3 个评价指标：用户兴趣，超链接，首页推荐。先使用层次分析法 [54]（层次结构如图24）确定各个指标的权。对引入的 3 个

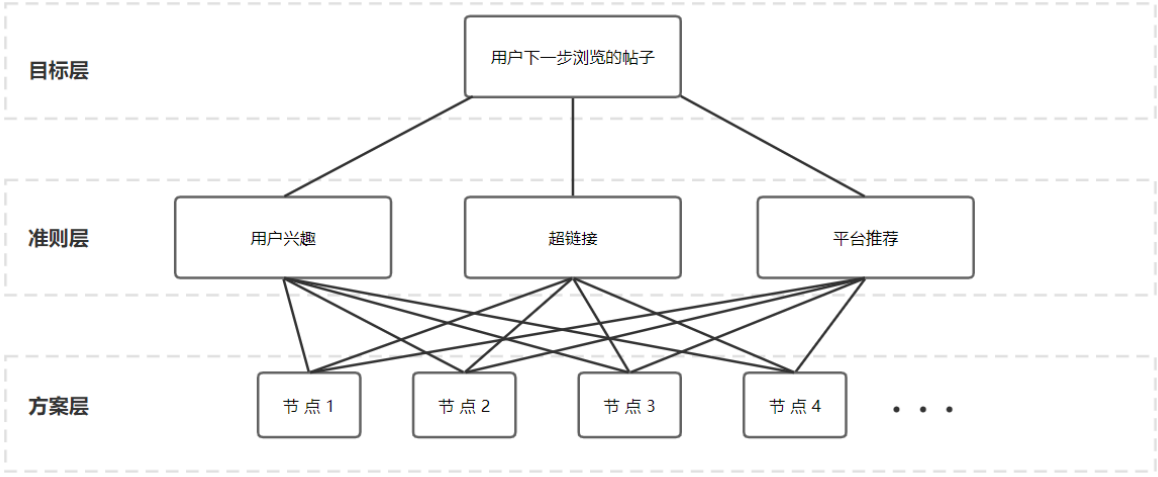


图 **24**　**TOPSIS** 层次结构图

评价指标（用户兴趣，超链接，首页推荐）使用层次分析法（层次结构如图24）确定各个指标的权重：将评价指标两两比较，通过 Santy 的 1-9 标度方法 [54]（如表5）构造出判断矩阵（如表6），该矩阵为一致矩阵，所以可以直接计算出各评价指标的权重为：

用户兴趣：$w_1 = \frac{4}{4+1+1} = \frac{2}{3}$；超链接：$w_2 = \frac{1}{4+1+1} = \frac{1}{6}$；平台推荐算法：$w_3 = \frac{1}{4+1+1} = \frac{1}{6}$

确定权重后，需要对各节点评分，这里首先说明一下三个评价指标的评分细节：

- 用户兴趣评分：分数为 k^n，k 为节点的入度，n 取决于用户的观点和帖子本身的观点，当两者的观点距离小于信任阈值 ϵ_2 时，n=3；当两者的观点距离在 ϵ_2 和 $2 \times \epsilon_2$ 之间时，n=2；当两者的观点距离大于 $2 \times \epsilon_2$ 时，n=1。
- 超链接评分：假设用户当前所在的节点为 A，如果有一条超链接 $A \to B$，则 B 的得分为 1，如果不存在 $A \to B$，则 B 的得分为 0。
- 首页推荐评分：假设平台参考用户过去几步内浏览的帖子，算出这些帖子所带观点的平均值 o，假设节点所带的观点为 o_i，那么各节点的得分就是 $|o_i - o|$。

根据以上评价指标的定义得出一个初始评分表，如表7。接着将该矩阵指标正向化，即将所有的评价指标转化为极大型指标（越大越好，此处首页推荐指标下的评分取倒数），如表8。由表8可以得到一个评分矩阵，如式49，然后再进行标准化，从而消除不

表 **5**　**Santy** 标度方法

标度	含义
1	同样重要性
3	稍微重要
5	明显重要
7	强烈重要
9	极端重要
2，4，6，8	上述两相邻判断的中值
倒数	A 和 B 相比如果标度为 3，那么 B 和 A 相比就是 1/3

表 **6**　判断矩阵

	用户兴趣	超链接	平台推荐
用户兴趣	1	4	4
超链接	$\frac{1}{4}$	1	1
平台推荐	$\frac{1}{4}$	1	1

同指标量纲的影响。

$$\begin{bmatrix} 27 & 0 & 5 \\ 64 & 0 & 2.94 \\ 16 & 1 & 10 \\ 7 & 0 & 0.83 \\ \cdots & \cdots & \cdots \end{bmatrix} \tag{49}$$

这里阐述一下该过程，假设有 n 个要评价的对象（此处指的是节点），m 个已经正向化的评价指标（上述三个评价指标），其构成的正向化矩阵如式50。将矩阵（式50）的标准化矩阵记为 Z，Z 中的每一个元素如式51，然后计算每个节点的评分并归一化。

表 **7**　初始评分表

	用户兴趣	超链接	首页推荐
节点 1	27	0	0.2
节点 2	64	0	0.34
节点 3	16	1	0.1
节点 4	7	0	1.2
……	……	……	……

表 **8**　正向化评分表

	用户兴趣	超链接	首页推荐
节点 1	27	0	1/0.2=5
节点 2	64	0	1/0.34=2.94
节点 3	16	1	1/0.1=10
节点 4	7	0	1/1.2=0.83
……	……	……	……

接着将该表中的所有指标正向化处理，即将所有的评价指标转化为极大型指标（越大越好，此处首页推荐指标下的评分），如图 4

$$\begin{bmatrix} x_{11} & x_{12} & \dots & x_{1m} \\ x_{21} & x_{22} & \dots & x_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nm} \end{bmatrix} \tag{50}$$

$$z_{ij} = \frac{x_{ij}}{\sqrt{\sum_{i=1} n x_{ij}^2}} \tag{51}$$

对于有 n 个待评价对象，m 个评价指标，其标准化矩阵如式52。

$$Z = \begin{bmatrix} z_{11} & z_{12} & \cdots & z_{1m} \\ z_{21} & z_{22} & \cdots & z_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ z_{n1} & z_{n2} & \cdots & z_{nm} \end{bmatrix} \tag{52}$$

定义最大值 z^+：

$$\begin{aligned} z^+ &= (z_1^+, z_2^+, \ldots, z_m^+) \\ &= (max\{z_{11}, z_{21}, \ldots, z_{n1}\}, max\{z_{12}, z_{22}, \ldots, z_{n2}\}, \ldots, max\{z_{1m}, z_{2m}, \ldots, z_{nm}\}) \end{aligned}$$

定义最小值 z^-：

$$\begin{aligned} z^- &= (z_1^-, z_2^-, \ldots, z_m^-) \\ &= (min\{z_{11}, z_{21}, \ldots, z_{n1}\}, min\{z_{12}, z_{22}, \ldots, z_{n2}\}, \ldots, min\{z_{1m}, z_{2m}, \ldots, z_{nm}\}) \end{aligned}$$

定义第 i（$i = 1, 2, \ldots, n$）个评价对象与最大值的距离 $D_i^+ = \sqrt{\sum_{j=1}^m w_j (z_j^+ - z_{ij})^2}$

定义第 i（$i = 1, 2, \ldots, n$）个评价对象与最小值的距离 $D_i^- = \sqrt{\sum_{j=1}^m w_j (z_j^- - z_{ij})^2}$

那么，我们可以计算得出第 i 个评价对象未归一化的得分：$S_i = \frac{D_i^-}{D_i^+ + D_i^-}$，显然有 $0 \leqslant S_i \leqslant 1$。而且 S_i 越大，D_i^+ 越大，即越接近最大值。若使用层次分析法给 m 个评价指标确定权重，那么权重 w_j 满足式53.

$$\sum_{j=1}^{m} w_j = 1 \tag{53}$$

从而得到如表9所示的评分结果 再将各节点的得分归一化，即每个节点的得分除各节点

表 **9**　评分结果

节点 1	节点 2	节点 3	节点 4	节点 5	节点 6	节点 7	节点 8	…
4	6	0	4	7	8	2	4	…

得分的总和，结果如表10(表中 n 代表总结点数)。以此评分作为用户向各节点的转移概率。特别说明：

1. 假设用户当前处于 A 节点，那么此节点评分默认为 0，以此表示用户下一步一定会离开当前所在的节点。

2. 假设用户当前在 A 节点，即使没有超链接 $A \to B$，用户下一步也有可能去 B，因为 B 节点的评分不一定为 0.

表 **10** 归一化的评分结果

节点 1	节点 2	节点 3	节点 4	节点 5	节点 6	节点 7	节点 8	…
4/n	6/n	0/n	4/n	7/n	8/n	2/n	4/n	…

得到用户向各节点的转移概率后，我们接着让用户在网络中游走以观察茧房形成的情况。规定用户每天游走一定的步数，用户游走到某一节点即视为受到该节点所带观点的影响，影响方式同式47。同时，引入用户活跃度 a，当观点相近时 $\zeta_1 = \frac{1}{\pi}arctan(a)$；当观点超出信任阈值时，$\zeta_2 = \frac{0.2}{\pi}arctan(a)$。且用户活跃度越高，趋同系数越高。每天都令一定数量的用户同时在网络中做游走，当某一步有多个用户游走到同一节点，则再计入用户之间的相互影响，影响方式同47。这里阐明一下几个定义（含假设）：

- 信息茧房深度的定义：某一天陷入信息茧房的用户的占比。
- 话题吸引度：决定每天增加的帖子数量，吸引度越高就意味着会有更多的人再网络上发帖子，每天新增的帖子的数量服从高斯分布，随着新的帖子加入网络，无标度网络逐逐渐形成，信息茧房深度加深。
- 用户心理：用户倾向于浏览观点相近的帖子以及热门的帖子，这一心理可以用前文用户兴趣评分定义中的 k^n 这一评价指标来反映。
- 用户活跃度：网络中用户活动越频繁（包括点赞、评论、转发等操作），用户活跃度越高，不同用户间发生观点交换的概率越大，因而趋同系数越大，即观点受到的影响越大。

我们根据上述过程设计模拟程序，并采取单一变量分析法分析单独每个影响因素对茧房深度（IC depth）的影响。模拟结果表明，用户活跃度越高，茧房深度越大，如图25. 这个结果可以这样理解：用户活跃度越高，趋同系数越高，有利于网络用户与与其具有相似或相同观点的用户产生联系，从而形成一个由具有相似或相同观点、偏好的社区，即信息茧房；前文定义平台推荐算法的影响分析——平台根据用户浏览浏览历史，算出用户历史浏览帖子所带观点的平均值，得出这个平均值和网络中各节点所带观点的差距，将这个差距取倒数得到各节点的首页推荐分数，从而来确定用户下一步会因平台推荐算法而浏览到的帖子，这里我们设置平台根据用户前不同步数前的浏览历史来推荐，得到相应的 IC depth，结果如图26. 由图可知，若平台记录用户的浏览步数越少，那么 IC

depth 越大，这可以这样理解：平台记录的浏览历史越多，对用户的兴趣爱好把握的越全面，形成的用户心理画像越真实，从而可以为用户推荐更多不同类型（不同观点）的内容，这样可以避免用户陷入某些同质性的话题中，即信息茧房；对于不同用户间的影响，我们定义不同用户间相互作用规则来看最终形成茧房的程度，分别有三种规则：不允许用户之间互相作用、只允许观点在信任阈值内的用户相互作用、允许超出信任阈值的用户以一个较低的趋同系数相互作用，结果如图27. 由图27可知，不允许用户间发生相互作用时茧房深度最大，而允许用户间发生相互作用时茧房深度会降低，对于该结果可以这样理解：用户互相作用就理解成用户游走到同一个节点时观点互相影响，用户的相互作用会导致信息茧房深度降低，因为用户了解到了更全面的信息，可以避免用户陷入片面的认知，从而可以降低用户陷入茧房的概率。

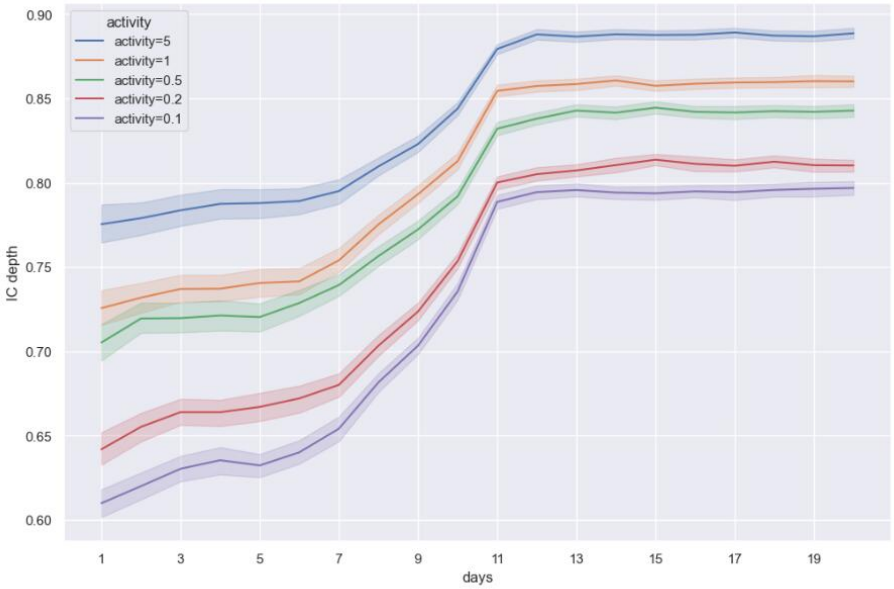


图 **25**　不同用户活跃度 **(activity)** 下茧房深度 **(IC depth)** 随时间的演化结果

5.3 问题三分析与求解

5.3.1 破除尖叫效应的策略

由 PageRank-SIR-RUCM 模型下调节度大的初始感染节点个数 n 与话题固有传播概率 λ 的模拟结果图18可知，当度相对较大的节点作为初始感染节点时，且其传播话题的固有话题传播概率较大时，度大的初始感染节点越多，尖叫效应越明显。可以直观地理解度大的节点在社交网络中拥有大量粉丝，加上其发布的话题能够吸引人、博人眼球时，那么将会产生尖叫效应。因此，要破除尖叫效应就需要重点关注拥有大量粉丝的用户，监控他们传播的话题，当其传播话题存在博人眼球，获取流量嫌疑时，需要对其进行管控。

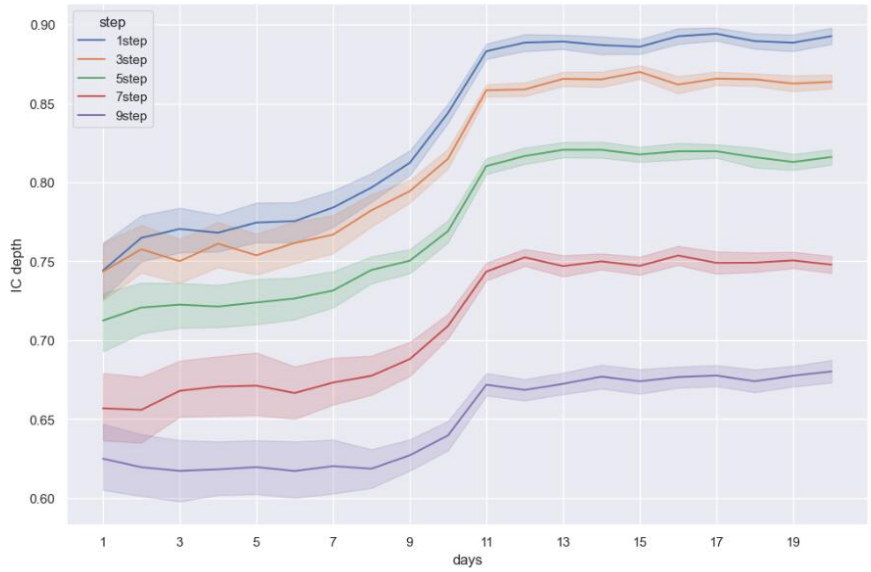


图 **26**　平台根据用户不同步数（**step**）前的浏览历史进行推荐下茧房深度 **(IC depth)** 随时间的演化结果

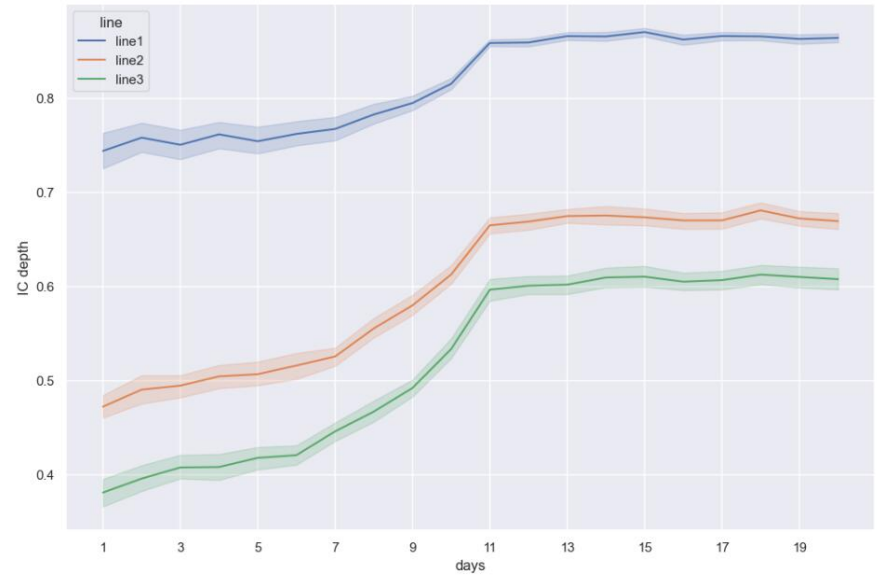


图 **27**　不同相互作用规则下茧房深度（**IC depth**）随时间的演化。**line1:** 不允许用户之间互相作用，**line2**：只允许观点在信任阈值内的用户相互作用，**line3**：允许超出信任阈值的用户以一个较低的趋同系数相互作用

5.3.2 破除回声室效应的策略

由自发对称性破缺模型我们可以定性地知道回声室的形成是因为网络发生自发对称性破缺，使得一些节点与主网络间发生相对隔离，从而导致相对孤立的那些节点不能或者不与主网络有联系。而通过相图（T, T^N）我们可以知道，处于回音室中的节点与

它的邻居的价值观念十分相似，从而导致他们只能接触到和自己价值取向、偏好相符的信息，加上他们所处的环境与主网络隔离出来，久而久之，这些处于回声室中的人就会陷入自己的世界，与外界脱钩，导致群体极化。因此，对于回音室效应的破除，个体是极为关键的。个体需要敢于走出自己的舒适区，接触新事物、接受不同的观点，并要丰开拓自己的视野，提升个人认知水平，这样才能破除回音室这把枷锁。

5.3.3 破规避信息茧房的策略

借助基于两个社区的观点动力学分析，我们知道，当一个社区内的用户对另一个社区内的其他用户的信任阈值越高，且社区内的趋同系数越大时，往往就会形成信息茧房。这就会导致用户同质化信息所包围，而对圈外则表现出排外心理，导致圈内同质化，圈外异质性。借助信息茧房深度影响因素分析模型，我们知道用户活跃度和平台推荐算法均会加深茧房深度，同时不同用户之间能否发生相互作用对茧房深度也有影响：若允许不同用户发生相互作用，一定程度上可以降低茧房深度。那么，就个人而言用户要警觉信息茧房，能开放包容地与持有不同观点的人相互交流可以避免自己陷入信息茧房。而对于社交媒体平台则要在充分了解用户的个性化需求的同时不能一味地迎合用户，一味地推荐类似的内容给用户会导致用户陷入信息茧房。因此，社交媒体有义务推荐更为均衡内容给用户，而不是一味迎合用户偏好。

5.4 问题四分析与求解

信息茧房 (IC) 的概念最早是美国教授 Sunstein 在《信息乌托邦：众人如何生产知识》(Infotopia:how many minds produce knowledge)[55] 中提出的。Sunstein 认为，在 Web 2.0 网络环境下，互联网企业为用户开发了个性化的信息推荐，使用户只会看到自己选择的、满足的、愉悦的信息，失去了深刻思考的能力 [55]。在当时的时代背景下，Sunstein 还认为，现实生活中的信息并不局限于推荐算法过滤的互联网信息，用户也不完全是生活在网络世界中的。他曾经承认，“在未来 10 年或更长的时间里，我的个人论点不应该被视为关于信息选择的个人特征的实证论点。”因此，IC 的概念应该被归类为一个假设，而不是一个被证明的理论。然而，随着互联网的普及，推荐系统推荐的互联网信息成为许多网民的主要信息源。由于推荐算法的可控性和可预见性有限，算法的逐渐复杂化以及商业保密的目的，企业的推荐系统机制至今仍是公众和学者的“黑箱”[56]。信息技术和服务的爆发式增长，特别是门户网站、推荐系统、搜索引擎和社交媒体的出现，使我们进入了一个信息丰富的世界。我们通过越来越多的信息源获取多样化信息，但人们普遍认为，我们往往会不知不觉地陷入信息茧房中，被有限的、带有偏见的信息包围 [55]。此外, 考虑混合网络信息、推荐系统和互联网用户的价值判断能力有限，许多专家学者担心这会导致社会分裂和冲突加剧，提出了弊端，并对 IC 形成的原因、IC 带来的危害和控制或减轻 IC 的形成的研究产生极大兴趣。影响 IC 的因素是多样的，但可以大

致被分为两类：

1. 积极选择：个体倾向于选择与志同道合的人建立关系，阅读和产生相似观点的文章，同时忽略不同的声音，形成回音室 [19, 20, 21, 57]。
2. 被动选择：搜索引擎和推荐系统会根据用户过去的记录向用户提供信息，产生过滤气泡，以及朋友推荐和好友推荐会缩小用户的网络导航范围 [58, 59, 60, 61, 62, 63]。

IC 的扩散可能回导致社会分裂、两极分化和极端主义的加剧，最终加剧割裂，威胁民主 [55, 64, 65]。虽然与 IC 的相关问题受到了广泛地调查和激烈地讨论 [19, 66, 67, 68, 69]，但关于 IC 的存在和影响的定量研究却很少。在前文我们已经分析了 IC 的形成机制和 IC depth 影响因素的分析。接下来，我们将关注点转向如何破除 IC。

为有效地分析“如何破除 IC”，我们采取自底向上的分析方法，结合文献 [70]，我们梳理了从信息资源整合到信息推送的全流程框架，如图28．并从网络用户的责任担当

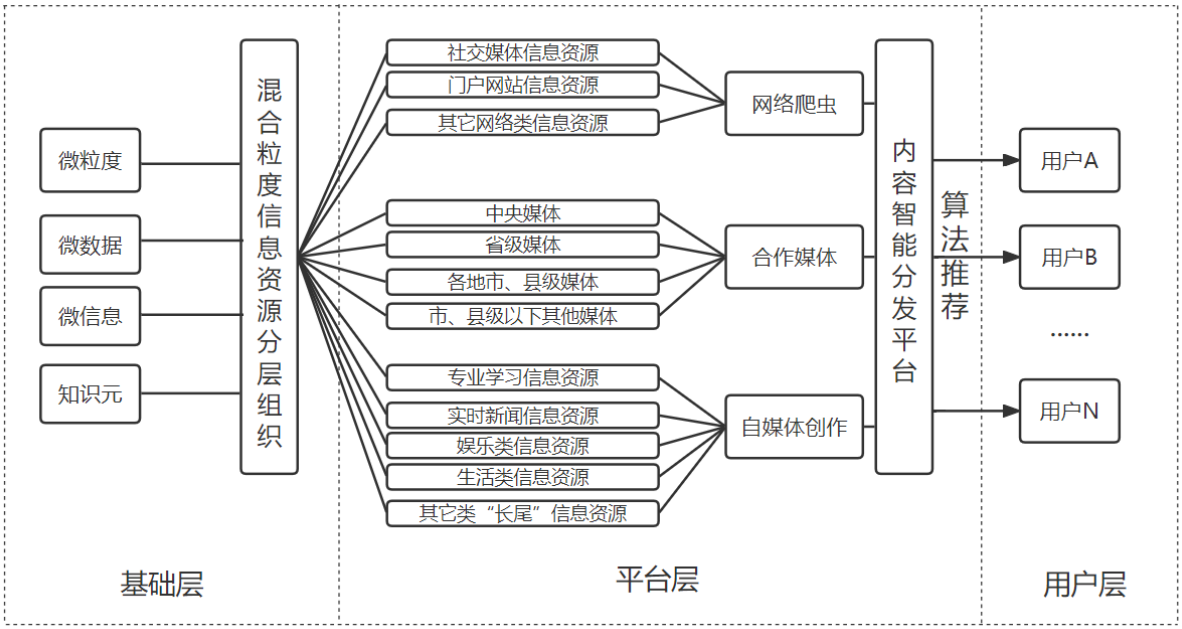


图 **28**　信息组织、传递、推送全流程自底向上结构

到国家社会层面的政府顶层设计提出合理建议。结合图22和图27我们可以知道若用户对自己所处的社区的其他用户信任阈值高，对其他社区信任阈值低的情况下更易于陷入信息茧房，那么这就意味从个人层面来说，用户应当走出自己的舒适区，以开放包容的心态接纳新信息与敢于和不同类型的人广泛交流，以及有着清醒的头脑甄别信息，可避免自己信息接触面受限，可以有效地避免自己陷入信息茧房。并且，用户即扮演信息传播过程中的信息接收者，同时也扮演参与者的双重角色，更是群体极化行为的主要参与者，个人应积极避免自身陷入信息茧房，要清晰地认识自己的网络活动是否会助长群体极化。同时，每个用户都要有学习提升个人素养的意识，这样才能有意识地对媒介信息进行判断、解读以及驾驭媒介信息，才能在享有信息时代的便利的同时也能突破信息茧

房的桎梏，与主流价值观相向而行，推动社会信息交流传递正向发展。但对于大部分用户来说，他们往往会不知不觉地就陷入信息茧房 [55]，这很大程度上是与现在人们获取信息的渠道（主要源于网络）和内容推荐系统有关。结合图26，我们知道平台的推荐算法对用户浏览历史记录分析的越多，对用户的偏好、心理所做的用户画像会更准确，从而做个性化推荐时能够推荐更多样性的内容，在一定程度上可以避免用户陷入信息茧房或陷入更深的信息茧房。但平台对用户的数据画像完全掌握并过度迎合用户的喜好时，也会致使用户陷入同质化的推荐内容之中，即陷入信息茧房。因而，主流媒体需要引领各个传媒平台，既要防止各大平台滥用算法，也要为用户宣传和普及利用好网络信息的知识，承担提升网名素养与传播主流价值观，防止社会群体极化的责任。主流媒体要发挥上述上述引领作用，需要提升自身传媒的影响力与吸引力，同时也要丰富主流媒体自身所传播的内容的多样性，这样才能即能影响互联网用户和带动其他传媒平台，也能为网民提供多种多样的信息，以避免用户陷入信息茧房之中。政府在治理、定制规则、监管等方面有绝对话语权，因而政府需要根据信息组织、传递、推送全流程自底向上结构（图28）制定从信息组织到平台监管到内容推送全流程合理标准，并对从信息组织、传递、推送全流程建立管理机制：

1. 在基础层信息组织阶段需要对上传到网络中的信息进行审核，避免会引起群体极化、社会分裂、背离主流价值观的信息进入大众视野。
2. 对于平台层需要对在各个平台上传播的信息进行监管，要求各个平台提供数据支持与协作配合识别、管控会引起信息茧房的信息，及时做出反应，同时要要求各个平台上传播的信息要具有均衡性、真实性、合法性。同时也要要求各大平台的推荐系统要做到能够满足用户个性化需求的同时，也要避免过度迎合用户，不断改进推荐算法，使得用户既能获取个性化信息，也能获取其他类型的信息，保证用户的信息来源具有多样性。
3. 对于用户层，要以教育宣传为主，不断提高用户的素养。倡导用户审慎辨别信息、理性讨论话题、辩证传播话题，扩展个人视野，避免陷入信息茧房。

六、 模型的改进

1. 平台推荐算法：可以建立起更加丰富的平台推荐机制，建立基于标签，用户，帖子，帖子属性，内容的多边网络 [71]，考虑更多的影响因素
2. 自发对称性破缺：文章中此物理模型并未定量地求解，只是定性地解释了回音室的产生机制，如果能定量地求出解，将是一大创新。

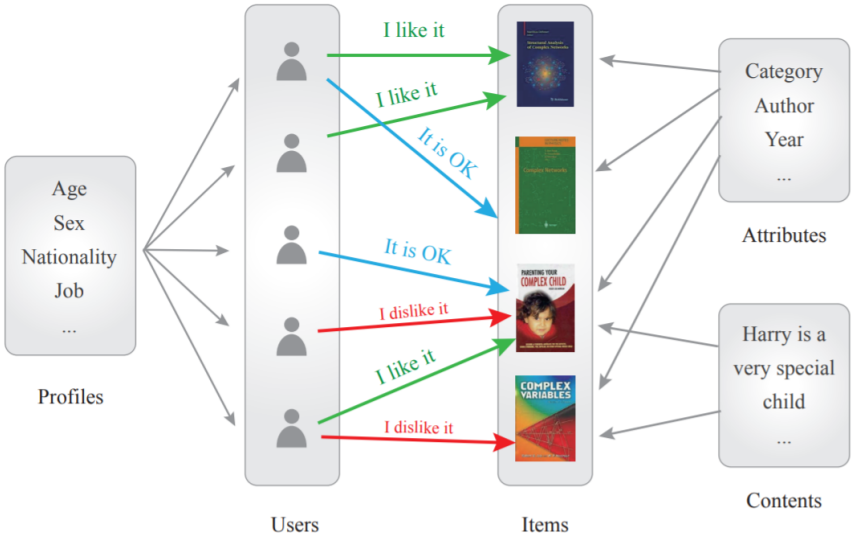


图 **29**　推荐系统示意图 [71]

七、 模型的优缺点

7.1 模型的优点

1. SIR-BASFN：在线社交网络是一种非均质无标度网络，本模型将用户作为网络中的节点，考虑了非均质网络的影响，建立 BASFN 可以很好地刻画社交网络，并以病毒传播模型 SIR 模型来刻画话题传播动力学，以此描述了社交网络上的话题传播过程。该模型具有简单明了的特点，且结果可以很好地定性描述话题在社交网络上的传播过程。
2. PageRank-SIR-RUCM：此模型将 SIR 和 RUCM 结合，同时再引入 PageRank 值描述不同用户 (节点) 在网络中的重要程度，能体现不同话题的传播能力差异，并考虑了用户心理、环境因素等许多复杂因子，且能够解释“尖叫效应”。另外，此模型还能同时描述社交网络中话题的传播过程和用户对话题的讨论过程，能较好的解释观点极化和中立共识。
3. BCM(Bounded Confidence Model)：能较好地解释观点达成中立共识的过程，即观点相近的用户在交互的过程中，他们的观点互相接近对方，达成共识。
4. RUCM（Rewiring with Unbounded Confidence Model）：基于传统的 BCM(Bounded Confidence Model) 模型做出了改进，一方面是允许观点差异较大的用户进行观点交互，另一方面允许网络中观点差异大的两个用户断开连接，以此来模拟社交网络中用户“吵架”和“取关”的行为，能够较好的解释观点极化过程。
5. 自发对称性破缺模型：从物理学的角度出发，借助描述在线用户动态的震荡模型框架下引入自发对称性破缺的概念，可以很好的阐述回声室效应的产生机制。

6. 信息茧房形成机制模型：通过构建两个社区，借助观点动力学可以探索两个社区之间的信息流通情况，从而揭示信息茧房的形成机制，该模型具有简单明了的特点。
7. 信息茧房深度影响因素分析模型：此模型依据帖子之间的超链接关系，以帖子作为节点构成网络，一方面考虑某一话题下帖子数量随着时间增长，网络结构将随之变化，另一方面模拟多个用户同时在网络中游走并发生观点交互的过程，较好的解释了信息茧房的加深。对于用户在“网上冲浪”刷帖的行为，引入了 TOPSIS 方法来确定用户在节点间游走的转移概率，以此描述用户对于不同帖子的喜好以及茧房的加深过程。

7.2 模型的缺点

1. SIR-BASFN：此模型未考虑不同话题传播能力的差异、未描述用户观点交互的过程、用户心理等因素对话题传播的影响。
2. PageRank-SIR-RUCM：对于平台推荐算法这一因素的考量太过简化。
3. BCM(Bounded Confidence Model)：社交网络中存在大量的“争吵”行为，BCM 默认了观点差异较大的用户之间不存在互动，与事实不符。
4. RUCM（Rewiring with Unbounded Confidence Model）：未描述话题在社交网络中的传播过程。
5. 自发对称性破缺模型：模型过于抽象，对于没有高等量子力学背景的人难以理解。
6. 信息茧房形成机制模型：不确定扩大到多个社区时的适用性。
7. 信息茧房深度影响因素分析模型：默认了用户游走到同一节点时必然发生观点交互，而事实上用户不一定发表评论，也不一定能看到对方的评论，也就是说用户之间不一定会发生观点交互。

参考文献

[1] 乙智. 网络平台的“尖叫效应”与“信息茧房”. 青年记者, 0(15):4–5, 2018.

[2] Q Vera Liao and Wai-Tat Fu. Can you hear me now? mitigating the echo chamber effect by source position indicators. In Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing, pages 184–196, 2014.

[3] 胡泳. “新词探讨：回声室效应”. 新闻与传播研究, 22(06):109–115, 2015.

[4] Matteo Cinelli, Gianmarco De Francisci Morales, Alessandro Galeazzi, Walter Quattrociocchi, and Michele Starnini. The echo chamber effect on social media. Proceedings of the National Academy of Sciences, 118(9):e2023301118, 2021.

[5] Kieron O'Hara and David Stevens. Echo chambers and online radicalism: Assessing the internet's complicity in violent extremism. Policy & Internet, 7(4):401–422, 2015.

[6] Siying Du and Steve Gregory. The echo chamber effect in twitter: does community polarization increase? In International workshop on complex networks and their applications, pages 373–378. Springer, 2016.

[7] Andrew Guess, Brendan Nyhan, Benjamin Lyons, and Jason Reifler. Avoiding the echo chamber about echo chambers. Knight Foundation, 2:1–25, 2018.

[8] Lei Hou, Xue Pan, Kecheng Liu, Zimo Yang, Jianguo Liu, and Tao Zhou. Information coons in online navigation. arXiv preprint arXiv:2109.06589, 2021.

[9] Jiyoung Woo, Jaebong Son, and Hsinchun Chen. An sir model for violent topic diffusion in social media. In Proceedings of 2011 IEEE International Conference on Intelligence and Security Informatics, pages 15–19, 2011.

[10] Jiyoung Woo and Hsinchun Chen. An event-driven sir model for topic diffusion in web forums. In 2012 IEEE International Conference on Intelligence and Security Informatics, pages 108–113, 2012.

[11] 何大韧，刘宗华，汪秉宏. 复杂系统与复杂网络. 北京：高等教育出版社, 2012.

[12] 孙玺菁，司守奎. 复杂网络算法与应用. 北京：国防工业出版社, 2015.

[13] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. science, 286(5439):509–512, 1999.

[14] 郭世泽，陆哲明. 复杂网络基础理论. 北京：科学出版社, 2012.

[15] 胡柯. 复杂网络上的传播动力学研究. Master's thesis, 湘潭大学, 2006.

[16] 刘自然. 加反馈机制的复杂网络动力学. Master's thesis, 湖南师范大学, 2007.

[17] 田蓓蓓. 复杂网络传播行为的元胞自动机模拟研究. Master's thesis, 上海大学, 2008.

[18] 周杰. 复杂系统中的信息传播研究. PhD thesis, 华东师范大学, 2008.

[19] Eytan Bakshy, Solomon Messing, and Lada A Adamic. Exposure to ideologically diverse news and opinion on facebook. Science, 348(6239):1130–1132, 2015.

[20] John Wihbey, Kenneth Joseph, and David Lazer. The social silos of journalism? twitter, news media and partisan segregation. New Media & Society, 21(4):815–835, 2019.

[21] Jiantao Hu, Qian-Ming Zhang, and Tao Zhou. Segregation in religion networks. EPJ Data Science, 8(1):6, 2019.

[22] Damon Centola. The spread of behavior in an online social network experiment. science, 329(5996):1194–1197, 2010.

[23] Xin Li, Lidong Bing, Wai Lam, and Bei Shi. Transformation networks for target-oriented sentiment classification. arXiv preprint arXiv:1805.01086, 2018.

[24] Michela Del Vicario, Antonio Scala, Guido Caldarelli, H Eugene Stanley, and Walter Quattrociocchi. Modeling confirmation bias and polarization. Scientific reports, 7(1): 1–9, 2017.

[25] Walter Quattrociocchi, Antonio Scala, and Cass R Sunstein. Echo chambers on facebook. Available at SSRN 2795110, 2016.

[26] Alessandro Bessi, Mauro Coletto, George Alexandru Davidescu, Antonio Scala, Guido Caldarelli, and Walter Quattrociocchi. Science vs conspiracy: Collective narratives in the age of misinformation. PloS one, 10(2):e0118093, 2015.

[27] Alessandro Bessi, Fabio Petroni, Michela Del Vicario, Fabiana Zollo, Aris Anagnostopoulos, Antonio Scala, Guido Caldarelli, and Walter Quattrociocchi. Viral misinformation: The role of homophily and polarization. 05 2015.

[28] Michela Del Vicario, Gianna Vivaldo, Alessandro Bessi, Fabiana Zollo, Antonio Scala, Guido Caldarelli, and Walter Quattrociocchi. Echo chambers: Emotional contagion and group polarization on facebook. Scientific reports, 6(1):1–12, 2016.

[29] Cass R Sunstein. The law of group polarization, 10 jpol. PHIL, 175:179–80, 2002.

[30] Raymond S Nickerson. Confirmation bias: A ubiquitous phenomenon in many guises. Review of general psychology, 2(2):175–220, 1998.

[31] Damon Centola and Andrea Baronchelli. The spontaneous emergence of conventions: An experimental study of cultural evolution. Proceedings of the National Academy of Sciences, 112(7):1989–1994, 2015.

[32] Michela Del Vicario, Alessandro Bessi, Fabiana Zollo, Fabio Petroni, Antonio Scala, Guido Caldarelli, H Eugene Stanley, and Walter Quattrociocchi. The spreading of misinformation online. Proceedings of the National Academy of Sciences, 113(3):554–559, 2016.

[33] Michela Del Vicario, Antonio Scala, Guido Caldarelli, H Eugene Stanley, and Walter Quattrociocchi. Modeling confirmation bias and polarization. Scientific reports, 7(1): 1–9, 2017.

[34] Guillaume Deffuant, David Neau, Frederic Amblard, and Gérard Weisbuch. Mixing beliefs among interacting agents. Advances in Complex Systems, 3(01n04):87–98, 2000.

[35] Rainer Hegselmann, Ulrich Krause, et al. Opinion dynamics and bounded confidence models, analysis, and simulation. Journal of artificial societies and social simulation, 5 (3), 2002.

[36] Jan Lorenz. Continuous opinion dynamics under bounded confidence: A survey. International Journal of Modern Physics C, 18(12):1819–1838, 2007.

[37] Kathleen Hall Jamieson and Joseph N Cappella. Echo chamber: Rush Limbaugh and the conservative media establishment. Oxford University Press, 2008.

[38] R Kelly Garrett. Echo chambers online?: Politically motivated selective exposure among internet news users. Journal of computer-mediated communication, 14(2):265–285, 2009.

[39] Kiran Garimella, Gianmarco De Francisci Morales, Aristides Gionis, and Michael Mathioudakis. Political discourse on social media: Echo chambers, gatekeepers, and the price of bipartisanship. In Proceedings of the 2018 world wide web conference, pages 913–922, 2018.

[40] Wesley Cota, Silvio C Ferreira, Romualdo Pastor-Satorras, and Michele Starnini. Quantifying echo chamber effects in information spreading over political communication networks. EPJ Data Science, 8(1):1–13, 2019.

[41] Matteo Cinelli, Gianmarco De Francisci Morales, Alessandro Galeazzi, Walter Quattrociocchi, and Michele Starnini. The echo chamber effect on social media. Proceedings of the National Academy of Sciences, 118(9):e2023301118, 2021.

[42] S Galam. The random symmetry breaking choice. Sec, 6(2):108–109, 2016.

[43] Masaki Aida, Ayako Hashizume, Chisa Takano, and Masayuki Murata. Polarization model of online social networks based on the concept of spontaneous symmetry breaking. In 2020 32nd International Teletraffic Congress (ITC 32), pages 106–113, 2020.

[44] Aida Masaki and Hashizume Ayako. Modeling of online echo-chamber effect based on the concept of spontaneous symmetry breaking. IEICE Proceeding Series, 63:IB3–4, 2020.

[45] Marta Arias, Ramon Ferrer-i Cancho, and Argimiro Arratia. Introduction to network dynamics. 2020.

[46] Masaki Aida, Chisa Takano, and Masaki Ogura. On the fundamental equation of user dynamics and the structure of online social networks. In International Conference on Network Science, pages 155–170. Springer, 2020.

[47] Yoshiki Kuramoto. Chemical oscillations, waves and turbulence. mineola, 2003.

[48] Kiran Garimella, Gianmarco De Francisci Morales, Aristides Gionis, and Michael Mathioudakis. Quantifying controversy on social media. ACM Transactions on Social Computing, 1(1):1–27, 2018.

[49] 许志源，唐维庸. 2016 美国大选所透射的“过滤气泡”现象与启示. 海外传媒, 2017 (16):54–56, 2017.

[50] Kiran Garimella, Gianmarco De Francisci Morales, Aristides Gionis, and Michael Mathioudakis. Reducing controversy by connecting opposing views. In Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, pages 81–90, 2017.

[51] Lei Hou, Xue Pan, Kecheng Liu, Zimo Yang, Jianguo Liu, and Tao Zhou. Information cocoons in online navigation. arXiv preprint arXiv:2109.06589, 2021.

[52] Mingge Xiong, Yu Wang, and Zhe Cheng. Research on modeling and simulation of information cocoon based on opinion dynamics. In 2021 The 9th International Conference on Information Technology: IoT and Smart City, pages 161–167, 2021.

[53] 李国彬朱道俊, 张文锋. 基于熵权和 topsis 法的山区 35kv 架空线路雷击风险评估. 电气技术, 23(08):23–30, 2022.

[54] 殷玮川冯柏盛. 基于层次分析法的城市交通网络可达性评价方法研究. 现代城市轨道交通, 2022(04):66–71, year=2022, publisher=SpringerOpen.

[55] Cass R Sunstein. Infotopia: How many minds produce knowledge. Oxford University Press, 2006.

[56] Lewis S C Nechushtai E. What kind of news gatekeepers do we want machines to be? filter bubbles, fragmentation, and the normative dimensions of algorithmic recommendations[j]. Computers in Human Behavior, 90:298–307, 2019.

[57] Mohsen Mosleh, Cameron Martel, Dean Eckles, and David G Rand. Shared partisanship dramatically increases social tie formation in a twitter field experiment. Proceedings of the National Academy of Sciences, 118(7):e2022761118, 2021.

[58] Tao Zhou, Zoltán Kuscsik, Jian-Guo Liu, Matúš Medo, Joseph Rushton Wakeling, and Yi-Cheng Zhang. Solving the apparent diversity-accuracy dilemma of recommender systems. Proceedings of the National Academy of Sciences, 107(10):4511–4515, 2010.

[59] Eli Pariser. The filter bubble: What the Internet is hiding from you. penguin UK, 2011.

[60] Natali Helberger, Kari Karppinen, and Lucia D' acunto. Exposure diversity as a design principle for recommender systems. Information, Communication & Society, 21(2):191–207, 2018.

[61] Luca Maria Aiello, Alain Barrat, Rossano Schifanella, Ciro Cattuto, Benjamin Markines, and Filippo Menczer. Friendship prediction and homophily in social media. ACM Transactions on the Web (TWEB), 6(2):1–33, 2012.

[62] Michael A Beam. Automating the news: How personalized news recommender system design choices impact news reception. Communication Research, 41(8):1019–1041, 2014.

[63] Jakob Ohme. Algorithmic social media use and its relationship to attitude reinforcement and issue-specific political participation–the case of the 2015 european immigration movements. Journal of Information Technology & Politics, 18(1):36–54, 2021.

[64] Natalie Jomini Stroud. Polarization and partisan selective exposure. Journal of communication, 60(3):556–576, 2010.

[65] Cass R Sunstein. Is social media good or bad for democracy. SUR-Int'l J. on Hum Rts., 27:83, 2018.

[66] Frederik Zuiderveen Borgesius, Damian Trilling, Judith Möller, Balázs Bodó, Claes H De Vreese, and Natali Helberger. Should we worry about filter bubbles? Internet Policy Review. Journal on Internet Regulation, 5(1), 2016.

[67] Andrew Guess, Brendan Nyhan, Benjamin Lyons, and Jason Reifler. Avoiding the echo chamber about echo chambers. Knight Foundation, 2:1–25, 2018.

[68] Axel Bruns. Are filter bubbles real? John Wiley & Sons, 2019.

[69] Gregory Eady, Jonathan Nagler, Andy Guess, Jan Zilinsky, and Joshua A Tucker. How many people live in political bubbles on social media? evidence from linked survey and twitter data. Sage Open, 9(1):2158244019832705, 2019.

[70] 王益成，王萍，王美月，张卫东. 运动信息视角下内容智能分发平台突破“信息茧房”策略研究. 情报理论与实践, 41(05):114–119, 2018.

[71] Linyuan Lü, Matúš Medo, Chi Ho Yeung, Yi-Cheng Zhang, Zi-Ke Zhang, and Tao Zhou. Recommender systems. Physics Reports, 519(1):1–49, 2012. ISSN 0370-1573. URL https://www.sciencedirect.com/science/article/pii/S0370157312000828. Recommender Systems.

附录 A SIR-BASFN 模型模拟–matlab 源程序

```
%请先打开第一段程序脚本
%------------------------------------------------------------------------------
function ax=ggplotAxes2D(varargin)
%
% @author:slandarer
%
% 参数说明:
% ----------------------------------------------------
% AxesTheme | 坐标区域风格     | 'gray'/'economist'/'wsj'/'own1'
% ColorOrder | 图形对象颜色序列 | 'default'/'none'/'npg'/'lancet'/'starterk'
%                                 'Set1'/'Set2'/'Set3'/'Dark2'/'own1'
% LegendStyle | 图例样式       | 'ggplot'/'own1'
% EdgeStyle | 轮廓样式         | 'none'/'gray'/'white'/'ori'

% ax.Legend.UserData.NewBkg 图例新背景
% ax.Legend.UserData.NewTitle 图例新标题
% ax.UserData.NewYTick(i) Y轴新标签

% 获取要处理的坐标区域=====================================================
if strcmp(get(varargin{1},'type'),'axes' )
    ax=varargin{1};
else
    ax=gca;
end
hold(ax,'on')



% default==================================================================
theme.AxesTheme='gray';
theme.ColorOrder='default';
theme.LegendStyle='ggplot';
theme.EdgeStyle='none';

%从可变长度变量中提取有用信息=============================================
for i=1:length(varargin)
    tempVar=varargin{i};
    if strcmp(tempVar,'AxesTheme')||strcmp(tempVar,'axesTheme')||strcmp(tempVar,'axestheme')
        theme.AxesTheme=varargin{i+1};
    end
    if strcmp(tempVar,'ColorOrder')||strcmp(tempVar,'colorOrder')||strcmp(tempVar,'colororder')
        theme.ColorOrder=varargin{i+1};
    end
    if
```

```
        strcmp(tempVar,'LegendStyle')||strcmp(tempVar,'legendStyle')||strcmp(tempVar,'legendstyle')
        theme.LegendStyle=varargin{i+1};
    end
    if strcmp(tempVar,'EdgeStyle')||strcmp(tempVar,'edgeStyle')||strcmp(tempVar,'edgestyle')
        theme.EdgeStyle=varargin{i+1};
    end
end


% 配色方案
switch theme.ColorOrder
    case 'none'
    case 'default'
        ax.ColorOrder=[0.9900 0.4500  0.4500
    0.8500   0.5600        0
    0.6400   0.6500        0
    0.2200   0.7100        0
         0   0.7500   0.4900
         0   0.7500   0.7700
         0   0.6900   0.9600
    0.5800   0.5600   1.0000
    0.9100   0.4200   0.9500
    1.0000   0.3800   0.7400];
    case 'npg'
        ax.ColorOrder=[0.9000 0.2900  0.2100
    0.3000   0.7300   0.8400
         0   0.6300   0.5300
    0.2400   0.3300   0.5300
    0.9500   0.6100   0.5000
    0.5200   0.5700   0.7100
    0.5700   0.8200   0.7600
    0.8600        0        0
    0.4900   0.3800   0.2800
    0.6900   0.6100   0.5200];
    case 'lancet'
        ax.ColorOrder=[  0    0.2700   0.5500
    0.9300        0        0
    0.2600   0.7100   0.2500
         0   0.6000   0.7100
    0.5700   0.3700   0.6200
    0.9900   0.6900   0.5700
    0.6800        0   0.1600
    0.6800   0.7100   0.7100
    0.1100   0.1000   0.1000];
    case 'starterk'
        ax.ColorOrder=[0.8000 0.0500       0
    0.3600   0.5300   0.8500
```

```
0.5200   0.7400        0
1.0000   0.8000        0
0.4900   0.5300   0.5600
     0   0.7100   0.8900
     0   0.6900   0.4000];
case 'Set1'
    ax.ColorOrder=[0.8900 0.1000  0.1100
0.2200   0.4900   0.7200
0.3000   0.6900   0.2900
0.6000   0.3100   0.6400
1.0000   0.5000        0
1.0000   1.0000   0.2000
0.6500   0.3400   0.1600
0.9700   0.5100   0.7500
0.6000   0.6000   0.6000];
case 'Set2'
    ax.ColorOrder=[0.4000 0.7600  0.6500
0.9900   0.5500   0.3800
0.5500   0.6300   0.8000
0.9100   0.5400   0.7600
0.6500   0.8500   0.3300
1.0000   0.8500   0.1800
0.9000   0.7700   0.5800
0.7000   0.7000   0.7000];
case 'Set3'
    ax.ColorOrder=[0.5500 0.8300  0.7800
1.0000   1.0000   0.7000
0.7500   0.7300   0.8500
0.9800   0.5000   0.4500
0.5000   0.6900   0.8300
0.9900   0.7100   0.3800
0.7000   0.8700   0.4100
0.9900   0.8000   0.9000
0.8500   0.8500   0.8500
0.7400   0.5000   0.7400
0.8000   0.9200   0.7700
0.8300   0.8300   0.8300];
case 'Dark2'
    ax.ColorOrder=[0.1100 0.6200  0.4700
0.8500   0.3700   0.0100
0.4600   0.4400   0.7000
0.9100   0.1600   0.5400
0.4000   0.6500   0.1200
0.9000   0.6700   0.0100
0.6500   0.4600   0.1100
0.4000   0.4000   0.4000];
case 'own1'
```

```
        ax.ColorOrder=[0.8500 0.7100  0.8000
    0.3700   0.4400   0.6600
    0.7500   0.6900   0.8300
    0.3700   0.2200   0.5200
    0.8400   0.2500   0.5500
    0.7200   0.5200   0.5200
    0.6100   0.3800   0.6000
    0.0400   0.1400   0.2800
    1.0000   0.5800   0.3500
    0.9500   0.8900   0.7500];
end

% 部分plot scatter修饰
if false
childrenNum=length(ax.Children);
for i=1:childrenNum
    switch theme.EdgeStyle
        case 'none'
            EdgeColor=[];
        case 'gray'
            EdgeColor=[0.3 0.3 0.3];
        case 'white'
            EdgeColor=[0.96 0.96 0.96];
        case 'ori'
            EdgeColor=ax.ColorOrder(mod(i-1,size(ax.ColorOrder,1))+1,:).*0.5;
    end
    switch get(ax.Children(i),'type')
        case 'line'
            ax.Children(i).LineWidth=1.8;
            if ~isempty(EdgeColor)
                ax.Children(i).LineWidth=1.5;
                ax.Children(i).MarkerEdgeColor=EdgeColor;
                ax.Children(i).MarkerSize=8;
                ax.Children(i).MarkerFaceColor=ax.ColorOrder(mod(i-1,size(ax.ColorOrder,1))+1,:);
            end
        case 'scatter'
            if ~isempty(EdgeColor)
                ax.Children(i).MarkerFaceColor=ax.ColorOrder(mod(i-1,size(ax.ColorOrder,1))+1,:);
                ax.Children(i).LineWidth=1.5;
                ax.Children(i).MarkerEdgeColor=EdgeColor;
                ax.Children(i).SizeData=60;
            end
    end
end
end
```

```
if true
childrenNum=length(ax.Children);
lineSet=[];
scatterSet=[];
n=1;
for i=childrenNum:-1:1
    if strcmp(get(ax.Children(i),'type'),'scatter')
        scatterSet{n}=ax.Children(i);
        n=n+1;
    end
end
n=1;
for i=childrenNum:-1:1
    if strcmp(get(ax.Children(i),'type'),'line')
        lineSet{n}=ax.Children(i);
        n=n+1;
    end
end
for i=1:length(scatterSet)
    switch theme.EdgeStyle
        case 'none'
            EdgeColor=[];
        case 'gray'
            EdgeColor=[0.3 0.3 0.3];
        case 'white'
            EdgeColor=[0.96 0.96 0.96];
        case 'ori'
            EdgeColor=ax.ColorOrder(mod(i-1,size(ax.ColorOrder,1))+1,:).*0.5;

    end
    scatterSet{i}.MarkerEdgeColor=ax.ColorOrder(mod(i-1,size(ax.ColorOrder,1))+1,:);
    if ~isempty(EdgeColor)
        scatterSet{i}.LineWidth=1.5;
        scatterSet{i}.MarkerEdgeColor=EdgeColor;
        scatterSet{i}.SizeData=60;
        scatterSet{i}.MarkerFaceColor=ax.ColorOrder(mod(i-1,size(ax.ColorOrder,1))+1,:);
    end
end
for i=1:length(lineSet)
    switch theme.EdgeStyle
        case 'none'
            EdgeColor=[];
        case 'gray'
            EdgeColor=[0.3 0.3 0.3];
        case 'white'
            EdgeColor=[0.96 0.96 0.96];
        case 'ori'
```

```
            EdgeColor=ax.ColorOrder(mod(i-1,size(ax.ColorOrder,1))+1,:).*0.5;

        end
        lineSet{i}.LineWidth=1.8;
        lineSet{i}.Color=ax.ColorOrder(mod(i-1,size(ax.ColorOrder,1))+1,:);
        if ~isempty(EdgeColor)
            lineSet{i}.LineWidth=1.5;
            lineSet{i}.MarkerEdgeColor=EdgeColor;
            lineSet{i}.MarkerSize=8;
            lineSet{i}.MarkerFaceColor=ax.ColorOrder(mod(i-1,size(ax.ColorOrder,1))+1,:);
        end
    end
    end



    % legend 风格化
    if ~isempty(ax.Legend)
    switch theme.LegendStyle
        case 'ggplot'
            ax.Legend.FontSize=11;
            ax.Legend.Title.FontSize=14;
            ax.Legend.AutoUpdate='off';
            if ~isempty(regexpi(ax.Legend.Location,'out'))
                ax.Legend.Box='off';
                lgdPos=ax.Legend.Position;
                xyMin=[(lgdPos(1)-ax.Position(1))/ax.Position(3)*(ax.XLim(2)-ax.XLim(1))+ax.XLim(1),...
                    (lgdPos(2)-ax.Position(2))/ax.Position(4)*(ax.YLim(2)-ax.YLim(1))+ax.YLim(1)];
                xyMax=[(lgdPos(1)+lgdPos(3)-ax.Position(1))/ax.Position(3)*(ax.XLim(2)-ax.XLim(1))+ax.XLim(1),...
                    (lgdPos(2)+lgdPos(4)-ax.Position(2))/ax.Position(4)*(ax.YLim(2)-ax.YLim(1))+ax.YLim(1)];
                ax.Legend.Title.Visible='off';
                xyMin(1),xyMax(2)
                ax.Legend.UserData.NewTitle=text(ax,xyMin(1),xyMax(2),['
                    ',ax.Legend.Title.String],...
                    'FontSize',14,'VerticalAlignment','top','FontWeight','bold');
            else
                ax.Legend.Box='off';
                lgdPos=ax.Legend.Position;
                xyMin=[(lgdPos(1)-ax.Position(1))/ax.Position(3)*(ax.XLim(2)-ax.XLim(1))+ax.XLim(1),...
                    (lgdPos(2)-ax.Position(2))/ax.Position(4)*(ax.YLim(2)-ax.YLim(1))+ax.YLim(1)];
                xyMax=[(lgdPos(1)+lgdPos(3)-ax.Position(1))/ax.Position(3)*(ax.XLim(2)-ax.XLim(1))+ax.XLim(1),...
                    (lgdPos(2)+lgdPos(4)-ax.Position(2))/ax.Position(4)*(ax.YLim(2)-ax.YLim(1))+ax.YLim(1)];
                xDiff=(xyMax(1)-xyMin(1));
                yDiff=(xyMax(2)-xyMin(2));
                ax.Legend.UserData.NewBkg=rectangle(ax,'Position',[xyMin,xDiff,yDiff],'Curvature',0.2,...
                    'LineWidth',1.2,'EdgeColor',[0.39 0.41 0.39],'FaceColor',[1 1 1 .2]);
                %ax.Legend.Title.FontSize=14;
```

```
            ax.Legend.Title.Visible='off';
            ax.Legend.UserData.NewTitle=text(ax,xyMin(1),xyMax(2),['
                ',ax.Legend.Title.String],...
                'FontSize',14,'VerticalAlignment','top','FontWeight','bold');
        end
    case 'own1'
        ax.Legend.Color=[0.9412 0.9412 0.9412];
        ax.Legend.LineWidth=0.8;
        ax.Legend.FontSize=11;
end
end

% axes风格化
switch theme.AxesTheme
    case 'gray'
       ax.Parent.Color=[1 1 1];
       ax.Color=[0.9,0.9,0.9];
       ax.Box='off';
       grid(ax,'on');
       ax.TickDir='out';
       ax.GridColor=[1 1 1];
       ax.GridAlpha=1;
       ax.LineWidth=1.2;
       ax.XColor=[0.33,0.33,0.33];
       ax.YColor=[0.33,0.33,0.33];
       ax.TickLength=[0.015 0.025];
       plot(ax,[ax.XLim(2),ax.XLim(1),ax.XLim(1),ax.XLim(2),ax.XLim(2)],...
              [ax.YLim(2),ax.YLim(2),ax.YLim(1),ax.YLim(1),ax.YLim(2)],...
              'Color',[1 1 1],'LineWidth',2)
    case 'economist'
        ax.Parent.Color=[0.8400 0.8900 0.9200];
        ax.Color=[0.8400 0.8900 0.9200];
        ax.Parent.InvertHardcopy='off';
        ax.Box='off';
        ax.YGrid='on';
        ax.GridColor=[1 1 1];
        ax.GridAlpha=1;
        ax.LineWidth=1.2;
        ax.XColor=[0.33,0.33,0.33];
        ax.YColor='none';
        ax.TickLength=[0.015 0.025];
        for i=1:length(ax.YTick)
            ax.UserData.NewYTick(i)=...
               text(ax,ax.XLim(1)-ax.TickLength(1)/(ax.Position(3))*(ax.XLim(2)-ax.XLim(1)),...
               ax.YTick(i),ax.YTickLabel{i},'HorizontalAlignment','right','Color',[0.33,0.33,0.33]);
        end
    case 'wsj'
```

```
        ax.Parent.Color=[0.9700 0.9500 0.8900];
        ax.Color=[0.9700 0.9500 0.8900];
        ax.Parent.InvertHardcopy='off';
        ax.Box='off';
        ax.YGrid='on';
        ax.GridAlpha=1;
        ax.LineWidth=0.8;
        ax.YColor='none';
        ax.GridLineStyle=':';
        ax.TickLength=[0.015 0.025];
        for i=1:length(ax.YTick)
            ax.UserData.NewYTick(i)=...
                text(ax,ax.XLim(1)-ax.TickLength(1)/(ax.Position(3))*(ax.XLim(2)-ax.XLim(1)),...
                ax.YTick(i),ax.YTickLabel{i},'HorizontalAlignment','right','Color',[0.33,0.33,0.33]);
        end
    case 'own1'
        grid(ax,'on');
        ax.GridLineStyle='--';
        ax.LineWidth = 1;
end
end
%-------------------------------------------------------------------------------------------------------------
%打开第一段程序脚本后再执行第二段程序脚本
%需要设置的参数
N = 5000; %总节点个数
max_degree = 700;%预估最大的degree数
%建立无标度网络
m = 8;%每次加入的连线数
m0 = 60;  %初始的节点个数
A = zeros(N,N);%邻接矩阵
%先建立小型随机网络
%随机连边 1表示有边
for i=1:m0
    for j= (i+1):m0
        A(i,j)= round(rand()); %非1即0
        A(j,i) = A(i,j);
    end
end  %初始完成
DegreeInterval = zeros(1,N);
for new = m0+1:N
    %old vertice 度越大连接上的概率越大
    Degree = sum(A(1:new-1,1:new-1)); %每个顶点的度
    %制造出一个度的分布区间，模拟概率
    DegreeInterval(1) = Degree(1);
    for i=2:new-1
        DegreeInterval(i) = Degree(i)+DegreeInterval(i-1);
    end
```

```
    %连接 新节点 与 m个old节点
    AllDegree = sum(sum(A(1:new-1,1:new-1))); %整个图的总度
    for i = 1:m
        while 1
         %以概率从old节点中找到合适的顶点连接
         RandDegree = fix(AllDegree*rand()+1); %要与度区间包含RandDegree的顶点相连
         %找到 符合 要求的区间所属顶点
         Ans = find(RandDegree <= DegreeInterval(1:new-1));
         old = Ans(1);
         if A(new,old) == 0
            A(new,old) = 1;
            A(old,new) = 1;
            break;       %成功连接
         end
        end
    end
end
Degree = sum(A);
k_aver = sum(Degree)/N;
P = zeros(1,max_degree);%各种度的概率
% 演化
Num_k = zeros(1,max_degree);%度为k的节点的数量，从第m列开始填入
for i = m:max_degree
    Num_k(i) = size(find(i == Degree),2);
end
P = Num_k/N;
SIR =
     zeros(3,max_degree);%S在第一行，I在第二行，R在第三行，与矩阵Num_k对应,SIR(1,6)表示度为6的s节点数
%初始化状态
SIR(1,:) = Num_k(1,:);
%假设初始随机感染500个节点

for inital = 1:500
    while 1
        i = fix(max_degree*rand()+1);
        if SIR(1,i) ~= 0
            SIR(2,i) = SIR(2,i) + 1;
            SIR(1,i) = SIR(1,i) - 1;
            break;
        end
    end
end
step = 0;%计时器,从0时刻开始
density_S = zeros(1,50);%节点密度,预计1000步以内
density_I = zeros(1,50);
density_R = zeros(1,50);
density_S(1) = sum(SIR(1,:))/N;
```

```
density_I(1) = sum(SIR(2,:))/N;
density_R(1) = sum(SIR(3,:))/N;
%开始演化
while 1
    step = step + 1;
    last_SIR = SIR;%记录上一时刻的状态
    for i = m:max_degree
        %度为i的S节点演化
        n_s = last_SIR(2,:);
        n_s(i) = 0;
        effect_s = 0;
        if Num_k(i) == 0
        else
            for node_s = 1:max_degree
                effect_s = effect_s +
                    n_s(node_s)*(1-1/node_s)*(node_s*P(node_s)/k_aver)*last_SIR(1,i)/Num_k(i);
            end
        end
        SIR(1,i) = last_SIR(1,i) - effect_s;
        %度为i的R节点演化
        nki = last_SIR(2,:);
        nkr = last_SIR(3,:);
        nki(i) = 0;
        nkr(i) = 0;
        effect_r = 0;
        for node_r = 1:max_degree
            if Num_k(node_r) == 0

            else
                effect_r = effect_r +
                    (node_r*P(node_r)/k_aver)*(nki(node_r)+nkr(node_r))/Num_k(node_r);
            end
        end
        SIR(3,i) = last_SIR(3,i) + last_SIR(2,i)*(1/i + (1 - 1/i)*effect_r);%???????
        %度为i的I节点演化
        SIR(2,i) = Num_k(i) - SIR(1,i) - SIR(3,i);
    end
    density_S(step+1) = sum(SIR(1,:))/N;
    density_I(step+1) = sum(SIR(2,:))/N;
    density_R(step+1) = sum(SIR(3,:))/N;
    if sum(SIR(2,:)) < 1%结束条件
        for j = step+2:50
            density_S(j) = density_S(step+1);
            density_I(j) = density_I(step+1);
            density_R(j) = density_R(step+1);
        end
        break;
```

```
    end
end
t = 1:50;
hold on
plot(t,density_S,'X-b');
plot(t,density_I,'d-g');
plot(t,density_R,'o-r');
legend('S','I','R');
xlabel('time');
ylabel('node density');
ylim([0 1]);
hold off
```

附录 B PageRank-SIR-RUCM 模型仿真–matlab 源程序

```
%需要设置的参数
N = 2000; %总节点个数
m = 16;%每次加入的连线数，平均度约等于m,新加入节点应该是让旧节点来连接自己
m0 = 100;  %初始的节点个数
A = zeros(N,N);%邻接矩阵，A(i,j)表示i指向j，对列求和表示入度，对行求和表示出度
op = unifrnd(0,1,[1,N]);%使用周期性边界
e = 0.3;%用户之间的信任阈值
total = 100;%模拟多少天
sita = 0.5;%在线概率
u = 0.15;%趋同系数
q = 0.85;%pagerank跳转因子
h = 2;%外部社会加强因子
w = 0.1;%网络用户对消息的不敏感度，(0,1]
rho = 5;%信息传播增长速率相关系数
epsilo = 2;%免疫增长因子
lamda = 0.7;%信息的固有传播概率,(0,1)
%先建立无标度有向网络
%假设初始节点只要观点在信任阈值内就互相连接
for i = 1:m0
    for j = i+1:m0
        if (op(i) - op(j))>=-1&&(op(i) - op(j))<-0.5
            pj = -1;
        end
        if (op(i) - op(j))>=-0.5&&(op(i) - op(j))<=0.5
            pj = 0;
        end
        if (op(i) - op(j))>0.5&&(op(i) - op(j))<=1
            pj = 1;
        end
        if abs(op(i) - op(j)- pj ) < e
```

```
            A(i,j) = 1;
            A(j,i) = 1;
        end
    end
end
for new = m0+1:N
    %先找出所有和新节点观点相近的旧节点
    cnt = 1;
    pos = zeros(1,N);
    for i = 1:new-1
        if (op(new) - op(i))>=-1&&(op(new) - op(i))<-0.5
            p = -1;
        end
        if (op(new) - op(i))>=-0.5&&(op(new) - op(i))<=0.5
            p = 0;
        end
        if (op(new) - op(i))>0.5&&(op(new) - op(i))<=1
            p = 1;
        end
        if abs(op(new) - op(i)- p ) < e
           pos(cnt) = i;
           cnt = cnt + 1;
        end
    end
    %出度越大连接上的概率越大
    out_Degree = sum(A(1:new-1,1:new-1),2); %每个顶点的出度
    out_Degree = out_Degree + 1;%出度最小为1
    pos = pos(1:cnt-1);
    out_Degree = out_Degree(pos);
    %制造出一个出度的分布区间，模拟概率
    out_DegreeInterval = cumsum(out_Degree);
    %连接 新节点 与 m个old节点
    All_out_Degree = sum(out_Degree); %总出度
    for i = 1:m
        while 1
            %以概率从old节点中找到合适的顶点连接
            RandDegree = fix(All_out_Degree*rand()+1); %要与度区间包含RandDegree的顶点相连
            %找到 符合 要求的区间所属顶点
            Ans = find(RandDegree <= out_DegreeInterval);
            Ans = Ans(1);
            old = pos(Ans);
            if A(old,new) == 0
                A(old,new) = 1;
                break;       %成功连接
            end
        end
    end
```

```
end
vv = op;
op = zeros(total,N);%用户观点,横标为天数,t = 1代表初始状态
op(1,:) = vv;
state =
    zeros(total,N);%记录每个节点的状态，横标为天数，S态(未知者)：0，I态（感染者）：1，R态（免疫者）：2
time = zeros(1,N);%节点计时器，记录感染者的衰退时间

%初始化,观点每间隔0.2找出出度最大的节点并感染
out_Degree = sum(A,2);
initial1 = find((0 <= op(1,:)) + (0.2 > op(1,:)) - 1);
[initial,im] = max(out_Degree(initial1));
state(1,initial1(im)) = 1;
out_Degree = sum(A,2);
initial1 = find((0.2 <= op(1,:)) + (0.4 > op(1,:)) - 1);
[initial,im] = max(out_Degree(initial1));
state(1,initial1(im)) = 1;
initial1 = find((0.4 <= op(1,:)) + (0.6 > op(1,:)) - 1);
[initial,im] = max(out_Degree(initial1));
state(1,initial1(im)) = 1;
initial1 = find((0.6 <= op(1,:)) + (0.8 > op(1,:)) - 1);
[initial,im] = max(out_Degree(initial1));
state(1,initial1(im)) = 1;
initial1 = find((0.8 <= op(1,:)) + (1 >= op(1,:)) - 1);
[initial,im] = max(out_Degree(initial1));
state(1,initial1(im)) = 1;
% for inital = 1:5
%     while 1
%         i = fix(N*rand()+1);
%         if state(1,i) == 0
%             state(1,i) = 1;
%             break;
%         end
%     end
% end


for t = 2:total
    %感染的节点先对邻居发生作用，再考虑感染和免疫
    %S态用户的观点不受别人影响也不影响别人，I态受别人影响也影响别人，R态观点受影响但不影响别人
    %只有I态影响别人,找出所有I态
    I = find(1 == state(t-1,:));%当前所有I节点的位置
    effect = zeros(1,N);%当天所有节点受到的影响
    for pos_I = 1:length(I)
        %找出这个节点指向的所有I态和R态节点
        %I态R态节点位置
        pos_IR = find((1 == state(t-1,:)) + (2 == state(t-1,:)));
```

```
cc = pos_IR;
for kill = 1:length(pos_IR)
    if A(I(pos_I),pos_IR(kill)) == 0
        cc(kill) = -1;
    end
end
poss = find(-1 ~= cc);
pos_IR = cc(poss);
%此时的pos_IR是这个节点指向的所有I态和R态节点的位置
for target = 1:length(pos_IR)
    if (op(t-1,pos_IR(target)) - op(t-1,I(pos_I)))>=-1&&(op(t-1,pos_IR(target)) -
        op(t-1,I(pos_I)))<-0.5
        p = -1;
    end
    if (op(t-1,pos_IR(target)) - op(t-1,I(pos_I)))>=-0.5&&(op(t-1,pos_IR(target)) -
        op(t-1,I(pos_I)))<=0.5
        p = 0;
    end
    if (op(t-1,pos_IR(target)) - op(t-1,I(pos_I)))>0.5&&(op(t-1,pos_IR(target)) -
        op(t-1,I(pos_I)))<=1
        p = 1;
    end
    if abs(op(t-1,pos_IR(target)) - op(t-1,I(pos_I)) - p) < e
        effect(pos_IR(target)) = effect(pos_IR(target)) + u*(op(t-1,I(pos_I)) -
            op(t-1,pos_IR(target)));
    else
        effect(pos_IR(target)) = effect(pos_IR(target)) - u*(op(t-1,I(pos_I)) -
            op(t-1,pos_IR(target)) + p);
        %开始重线
        A(I(pos_I),pos_IR(target)) = 0;
        %先找出所有观点相近的节点
        cnt = 1;
        pos = zeros(1,N);
        for i = 1:N
            if (op(t-1,pos_IR(target)) - op(t-1,i))>=-1&&(op(t-1,pos_IR(target)) -
                op(t-1,i))<-0.5
                p = -1;
            end
            if (op(t-1,pos_IR(target)) - op(t-1,i))>=-0.5&&(op(t-1,pos_IR(target)) -
                op(t-1,i))<=0.5
                p = 0;
            end
            if (op(t-1,pos_IR(target)) - op(t-1,i))>0.5&&(op(t-1,pos_IR(target)) -
                op(t-1,i))<=1
                p = 1;
            end
            if abs(op(t-1,pos_IR(target)) - op(t-1,i) - p ) < e
```

```
                pos(cnt) = i;
                cnt = cnt + 1;
            end
        end
        %出度越大连接上的概率越大
        out_Degree = sum(A,2); %每个顶点的出度
        out_Degree = out_Degree + 1;%出度最小为1
        pos = pos(1:cnt-1);
        out_Degree = out_Degree(pos);
        %制造出一个出度的分布区间，模拟概率
        out_DegreeInterval = cumsum(out_Degree);
        %连接 新节点 与 1个old节点
        All_out_Degree = sum(out_Degree); %总出度
        for rrr = 1:10%最多尝试10次重线
            %以概率从old节点中找到合适的顶点连接
            RandDegree = fix(All_out_Degree*rand()+1); %要与度区间包含RandDegree的顶点相连
            %找到 符合 要求的区间所属顶点
            Ans = find(RandDegree <= out_DegreeInterval);
            Ans = Ans(1);
            old = pos(Ans);
            if A(old,pos_IR(target)) == 0
                A(old,pos_IR(target)) = 1;
                break;       %成功连接
            end
        end
    end
end
end
op(t,:) = op(t-1,:)+effect;
%观点交互后可能会超出范围
range1 = find(1 < op(t,:));
range2 = find(0 > op(t,:));
op(t,range1) = abs(op(t,range1) - 1);
op(t,range2) = abs(op(t,range2) + 1);

%观点交互结束，开始对每个考虑感染和免疫
%先计算出所有节点的pagerank值
B = A';
P = zeros(N,N);
r = sum(B,2);
for i = 1:N
    for j = 1:N
        P(i,j) = (1-q)/N + q*B(i,j)/r(i);%状态转移矩阵
    end
end
[x,y] = eigs(B',1);
x = x/sum(x);%各节点pagerank值
```

```
    %找出所有S态节点，考虑感染
    infect_S = find(0 == state(t-1,:));
    %S态节点的消息总量
    d = zeros(1,length(infect_S));
    for i = 1:length(d)
        neighbor = find(1 == A(:,infect_S(i)));
        neighbor = neighbor';
        for j = 1:length(neighbor)
            if state(t-1,neighbor(j)) == 1
                d(i) = d(i) + x(neighbor(j));
            end
        end
    end
    %各S态节点感染概率
    n = sita*(1 - (1 - lamda).^d);
    panduan = rand(1,length(n));
    panduan = n - panduan;
    infect = find(0 < panduan);
    gg = infect_S(infect);
    state(t,gg) = 1;


    %找出所有I态节点，考虑免疫
    immue_I = find(1 == state(t-1,:));
    state(t,immue_I) = 1;
    time(immue_I) = time(immue_I)+ 1;
    %各I态节点免疫概率
    a = w./(1+ exp(rho - epsilo*time(immue_I)));
    panduan = rand(1,length(a));
    panduan = a - panduan;
    immue = find(0 < panduan);
    gg = immue_I(immue);
    state(t,gg) = 2;

    %找出所有R态节点，继承
    immue_R = find(2 == state(t-1,:));
    state(t,immue_R) = 2;
end
subplot(1,2,1)
hold on
x = 1:total;
S = zeros(1,total);
for t = 1:total
    ii = find(0 == state(t,:));
    S(t) = length(ii)/N;
end
I = zeros(1,total);
```

```
for t = 1:total
    ii = find(1 == state(t,:));
    I(t) = length(ii)/N;
end
R = zeros(1,total);
for t = 1:total
    ii = find(2 == state(t,:));
    R(t) = length(ii)/N;
end
plot(x,S,'b');
plot(x,I,'g');
plot(x,R,'r');
legend('S(Susceptile)','I(Infected)','R(Remove)');
xlabel('step');
ylabel('node density');
ylim([0 1]);
title('RUCM :randomly infect 5 nodes');
hold off
%ggplotAxes2D([],'AxesTheme','gray','LegendStyle','ggplot','ColorOrder','Set2');

subplot(1,2,2)
r = find(2 == state(total,:));
op_R = op(total,r);
xmin=0;
xmax=1;
op_x = linspace(xmin,xmax,20);
op_y = hist(op_R,op_x);
op_y = op_y/length(r);
plot (op_x,op_y,'s-','LineWidth',2);
xlim([0 1]);
xlabel('opinion');
ylabel('PDF');
title('Probability density functions of final opinion with =0.15,u=0.1');
legend('RUCM');
% ggplotAxes2D([],'AxesTheme','gray','LegendStyle','ggplot','ColorOrder','Set2');
```

附录 C BCM 的 PDF 峰数计算–matlab 源程序

```
%需要设置的参数
       N = 2000; %总节点个数

       step = 10^7;%步数

       %建立无标度网络
       m = 5;%每次加入的连线数
```

```
        m0 = 5;  %初始的节点个数
        A = zeros(N,N);%邻接矩阵
        %先建立小型随机网络
        %防bug
        for i=1:m0
            for j= (i+1):m0
                A(i,j)= 1;
                A(j,i) = A(i,j);
            end
        end
        DegreeInterval = zeros(1,N);
        for new = m0+1:N
            %old vertice 度越大连接上的概率越大
            Degree = sum(A(1:new-1,1:new-1)); %每个顶点的度
            %制造出一个度的分布区间，模拟概率
            DegreeInterval(1) = Degree(1);
            for i=2:new-1
                DegreeInterval(i) = Degree(i)+DegreeInterval(i-1);
            end
            %连接 新节点 与 m个old节点
            AllDegree = sum(sum(A(1:new-1,1:new-1))); %整个图的总度
            for i = 1:m
                while 1
                    %以概率从old节点中找到合适的顶点连接
                    RandDegree = fix(AllDegree*rand()+1); %要与度区间包含RandDegree的顶点相连
                    %找到 符合 要求的区间所属顶点
                    Ans = find(RandDegree <= DegreeInterval(1:new-1));
                    old = Ans(1);
                    if A(new,old) == 0
                        A(new,old) = 1;
                        A(old,new) = 1;
                        break;       %成功连接
                    end
                end
            end
        end
num_peaks = zeros(50,50);
e_cnt = 0;%计数器
u_cnt = 0;
for e = 0.01:0.01:0.5
    e_cnt = e_cnt + 1;
    u_cnt = 0;
    for u = 0.01:0.01:0.5
        u_cnt = u_cnt + 1;
        x = rand(1,N);%初始观点
        %BCM
        for t = 1:step
```

```
            i = fix(N*rand() + 1);
            j = fix(N*rand() + 1);
            if A(i,j) == 1
                if (x(i) - x(j))>=-1&&(x(i) - x(j))<-0.5
                    p = -1;
                end
                if (x(i) - x(j))>=-0.5&&(x(i) - x(j))<=0.5
                    p = 0;
                end
                if (x(i) - x(j))>0.5&&(x(i) - x(j))<=1
                    p = 1;
                end
                if abs(x(i)-x(j)-p) < e
                    op_i = x(i);
                    op_j = x(j);
                    x(i) = x(i) + u*(op_j - op_i);
                    x(j) = x(j) + u*(op_i - op_j);
                end
            end
        end
        %累积分布图
        xmin=min(x);
        xmax=max(x);
        op_BCM =
            linspace(xmin,xmax,20);%将最大最小区间分成50个等分点(49等分),然后分别计算各个区间的个数
        yy_BCM=hist(x,op_BCM);%计算各个区间的个数
        yy_BCM=yy_BCM/length(x);%计算各个区间的个数
        %寻找峰值
        [peaks,locs] = findpeaks(yy_BCM,'minpeakheight',0.01);
        num_peaks(e_cnt,u_cnt) = size(locs,2);
    end
end
Pm = 0.01:0.01:0.5;
Pc = 0.01:0.01:0.5;
pcolor(Pm,Pc,num_peaks);
colorbar;
xlabel(' ');
ylabel(' ');
title('BCM');
```

附录 D RBCM 的 PDF 峰数计算–matlab 源程序

```
%需要设置的参数
        N = 2000; %总节点个数
```

```
        step = 4*10^6;%步数
        attempt = 90;%最多尝试重连次数
        %建立无标度网络
        m = 5;%每次加入的连线数
        m0 = 5;  %初始的节点个数
        A = zeros(N,N);%邻接矩阵
        %防bug
        for i=1:m0
            for j= (i+1):m0
                A(i,j)= 1;
                A(j,i) = A(i,j);
            end
        end
        DegreeInterval = zeros(1,N);
        for new = m0+1:N
            %old vertice 度越大连接上的概率越大
            Degree = sum(A(1:new-1,1:new-1)); %每个顶点的度
            %制造出一个度的分布区间，模拟概率
            DegreeInterval(1) = Degree(1);
            for i=2:new-1
                DegreeInterval(i) = Degree(i)+DegreeInterval(i-1);
            end
            %连接 新节点 与 m个old节点
            AllDegree = sum(sum(A(1:new-1,1:new-1))); %整个图的总度
            for i = 1:m
                cnt = 0;%计数器，防bug
                while 1

                    %以概率从old节点中找到合适的顶点连接
                    RandDegree = fix(AllDegree*rand()+1); %要与度区间包含RandDegree的顶点相连
                    %找到 符合 要求的区间所属顶点
                    Ans = find(RandDegree <= DegreeInterval);
                    old = Ans(1);

                    if A(new,old) == 0
                        A(new,old) = 1;
                        A(old,new) = 1;
                        break;       %成功连接
                    end

                end
            end
        end
num_peaks = zeros(50,50);
e_cnt = 0;%计数器
u_cnt = 0;
for e = 0.01:0.01:0.5
```

```
e_cnt = e_cnt + 1;
u_cnt = 0;
for u = 0.01:0.01:0.5
    u_cnt = u_cnt + 1;
    A1 = A;
 x = rand(1,N);%初始观点
  %一次抽取一对节点，使之互相作用，再决定是否重线，如果要重线，只对i节点重线，j节点不管。如此往复
    for s = 1:step
        i = fix(N*rand()+1);
        j = fix(N*rand()+1);
        if A1(i,j) == 1
            if (x(i) - x(j))>=-1&&(x(i) - x(j))<-0.5
                pj = -1;
            end
            if (x(i) - x(j))>=-0.5&&(x(i) - x(j))<=0.5
                pj = 0;
            end
            if (x(i) - x(j))>0.5&&(x(i) - x(j))<=1
                pj = 1;
            end

            if (x(j) - x(i))>=-1&&(x(j) - x(i))<-0.5
                pi= -1;
            end
            if (x(j) - x(i))>=-0.5&&(x(j) - x(i))<=0.5
                pi = 0;
            end
            if (x(j) - x(i))>0.5&&(x(j) - x(i))<=1
                pi = 1;
            end
            %判断如何相互作用
            if abs(x(i) - x(j)- pj ) >= e

                A1(i,j) = 0;
                A1(j,i) = 0;
                %重连i
                while 1
                    Degree = sum(A1);%各节点的度
                    DegreeInterval(1) = Degree(1);
                    for c=2:N
                        DegreeInterval(c) = Degree(c)+DegreeInterval(c-1);
                    end
                    AllDegree = sum(Degree);
                    RandDegree = fix(AllDegree*rand()+1);
                    Ans = find(RandDegree <= DegreeInterval(1:N));%指出位置
                    k = Ans(1);
                    if A1(i,k) == 0
```

```
                        A1(i,k) = 1;
                        A1(k,i) = 1;
                        break;
                    end
                end

            else
                op_i = x(i);
                op_j = x(j);
                x(i) = x(i) + u*(op_j - op_i);
                x(j) = x(j) + u*(op_i - op_j);
            end
        end
    end
    %累积分布图
    xmin=min(x);
    xmax=max(x);
    op_RBCM =
        linspace(xmin,xmax,20);%将最大最小区间分成100个等分点(49等分),然后分别计算各个区间的个数
    yy_RBCM=hist(x,op_RBCM);%计算各个区间的个数
    yy_RBCM=yy_RBCM/length(x);%计算各个区间的个数

    %寻找峰值
    [peaks,locs] = findpeaks(yy_RBCM,'minpeakheight',0.02);
    num_peaks(e_cnt,u_cnt) = size(locs,2);

  end
end
Pm = 0.01:0.01:0.5;
Pc = 0.01:0.01:0.5;
pcolor(Pm,Pc,num_peaks);
colorbar;
xlabel(' ');
ylabel(' ');
title('RBCM');
```

附录 E UCM 的 PDF 峰数计算–matlab 源程序

```
%需要设置的参数
    N = 2000; %总节点个数

    step = 10^7;%步数

    %建立无标度网络
    m = 5;%每次加入的连线数
```

```
        m0 = 5;  %初始的节点个数
        A = zeros(N,N);%邻接矩阵
        %防bug
        for i=1:m0
            for j= (i+1):m0
                A(i,j)= 1;
                A(j,i) = A(i,j);
            end
        end
        DegreeInterval = zeros(1,N);
        for new = m0+1:N
            %old vertice 度越大连接上的概率越大
            Degree = sum(A(1:new-1,1:new-1)); %每个顶点的度
            %制造出一个度的分布区间，模拟概率
            DegreeInterval(1) = Degree(1);
            for i=2:new-1
                DegreeInterval(i) = Degree(i)+DegreeInterval(i-1);
            end
            %连接 新节点 与 m个old节点
            AllDegree = sum(sum(A(1:new-1,1:new-1))); %整个图的总度
            for i = 1:m
                while 1
                    %以概率从old节点中找到合适的顶点连接
                    RandDegree = fix(AllDegree*rand()+1); %要与度区间包含RandDegree的顶点相连
                    %找到 符合 要求的区间所属顶点
                    Ans = find(RandDegree <= DegreeInterval(1:new-1));
                    old = Ans(1);
                    if A(new,old) == 0
                        A(new,old) = 1;
                        A(old,new) = 1;
                        break;       %成功连接
                    end
                end
            end
        end
num_peaks = zeros(50,50);
e_cnt = 0;%计数器
u_cnt = 0;
for e = 0.01:0.01:0.5
    e_cnt = e_cnt + 1;
    u_cnt = 0;
    for u = 0.01:0.01:0.5
        u_cnt = u_cnt + 1;
         x = rand(1,N);%初始观点
        A1 = A;
        %UCM
        for t = 1:step
```

```
        i = fix(N*rand() + 1);
        j = fix(N*rand() + 1);
        if A1(i,j) == 1
            if (x(i) - x(j))>=-1&&(x(i) - x(j))<-0.5
                pj = -1;
            end
            if (x(i) - x(j))>=-0.5&&(x(i) - x(j))<=0.5
                pj = 0;
            end
            if (x(i) - x(j))>0.5&&(x(i) - x(j))<=1
                pj = 1;
            end

            if (x(j) - x(i))>=-1&&(x(j) - x(i))<-0.5
                pi= -1;
            end
            if (x(j) - x(i))>=-0.5&&(x(j) - x(i))<=0.5
                pi = 0;
            end
            if (x(j) - x(i))>0.5&&(x(j) - x(i))<=1
                pi = 1;
            end
            if abs(x(i)-x(j)-pj) < e
                op_i = x(i);
                op_j = x(j);
                x(i) = x(i) + u*(op_j - op_i);
                x(j) = x(j) + u*(op_i - op_j);
            else
                op_i = x(i);
                op_j = x(j);
                x(i) = x(i) - u*(op_j - op_i - pi);
                x(j) = x(j) - u*(op_i - op_j - pj);
                %xi和xj有可能超出范围
                if x(i)<0
                    x(i) = abs(x(i)+1);
                end
                if x(i)>1
                    x(i) = abs(x(i)-1);
                end
                if x(j)<0
                    x(j) = abs(x(j)+1);
                end
                if x(j)>1
                    x(j) = abs(x(j)-1);
                end
            end
        end
```

```
        end
        %累积分布图
        xmin=min(x);
        xmax=max(x);
        op_UCM =
            linspace(xmin,xmax,20);%将最大最小区间分成50个等分点(49等分),然后分别计算各个区间的个数
        yy_UCM=hist(x,op_UCM);%计算各个区间的个数
        yy_UCM=yy_UCM/length(x);%计算各个区间的个数
        %寻找峰值
        [peaks,locs] = findpeaks(yy_UCM,'minpeakheight',0.01);
        num_peaks(e_cnt,u_cnt) = size(locs,2);
    end
end
Pm = 0.01:0.01:0.5;
Pc = 0.01:0.01:0.5;
pcolor(Pm,Pc,num_peaks);
colorbar;
xlabel(' ');
ylabel(' ');
title('UCM');
```

附录 F RUCM 的 PDF 峰数计算–matlab 源程序

```
%需要设置的参数
        N = 2000; %总节点个数

        step = 4*10^6;%步数
        attempt = 90;%最多尝试重连次数
        %建立无标度网络
        m = 5;%每次加入的连线数
        m0 = 5;  %初始的节点个数
        A = zeros(N,N);%邻接矩阵
        %防bug
        for i=1:m0
            for j= (i+1):m0
                A(i,j)= 1;
                A(j,i) = A(i,j);
            end
        end
        DegreeInterval = zeros(1,N);
        for new = m0+1:N
            %old vertice 度越大连接上的概率越大
            Degree = sum(A(1:new-1,1:new-1)); %每个顶点的度
            %制造出一个度的分布区间，模拟概率
            DegreeInterval(1) = Degree(1);
```

```
            for i=2:new-1
                DegreeInterval(i) = Degree(i)+DegreeInterval(i-1);
            end
            %连接 新节点 与 m个old节点
            AllDegree = sum(sum(A(1:new-1,1:new-1))); %整个图的总度
            for i = 1:m
                cnt = 0;%计数器，防bug
                while 1

                    %以概率从old节点中找到合适的顶点连接
                    RandDegree = fix(AllDegree*rand()+1); %要与度区间包含RandDegree的顶点相连
                    %找到 符合 要求的区间所属顶点
                    Ans = find(RandDegree <= DegreeInterval);
                    old = Ans(1);

                    if A(new,old) == 0
                        A(new,old) = 1;
                        A(old,new) = 1;
                        break;       %成功连接
                    end

                end
            end
        end
num_peaks = zeros(50,50);
e_cnt = 0;%计数器
u_cnt = 0;
for e = 0.01:0.01:0.5
    e_cnt = e_cnt + 1;
    u_cnt = 0;
    for u = 0.01:0.01:0.5
        u_cnt = u_cnt + 1;
        A1 = A;
     x = rand(1,N);%初始观点
     %一次抽取一对节点，使之互相作用，再决定是否重线，如果要重线，只对i节点重线，j节点不管。如此往复
        for s = 1:step
            i = fix(N*rand()+1);
            j = fix(N*rand()+1);
            if A1(i,j) == 1
                if (x(i) - x(j))>=-1&&(x(i) - x(j))<-0.5
                    pj = -1;
                end
                if (x(i) - x(j))>=-0.5&&(x(i) - x(j))<=0.5
                    pj = 0;
                end
                if (x(i) - x(j))>0.5&&(x(i) - x(j))<=1
                    pj = 1;
```

```
end

if (x(j) - x(i))>=-1&&(x(j) - x(i))<-0.5
    pi= -1;
end
if (x(j) - x(i))>=-0.5&&(x(j) - x(i))<=0.5
    pi = 0;
end
if (x(j) - x(i))>0.5&&(x(j) - x(i))<=1
    pi = 1;
end
%判断如何相互作用
if abs(x(i) - x(j)- pj ) >= e
    op_i = x(i);
    op_j = x(j);
    x(i) = x(i) - u*(op_j - op_i - pi);
    x(j) = x(j) - u*(op_i - op_j - pj);
    %作用完后xi和xj可能会超出范围
    if x(i)<0
        x(i) = abs(x(i)+1);
    end
    if x(i)>1
        x(i) = abs(x(i)-1);
    end
    if x(j)<0
        x(j) = abs(x(j)+1);
    end
    if x(j)>1
        x(j) = abs(x(j)-1);
    end
    A1(i,j) = 0;
    A1(j,i) = 0;
    %重连i
    while 1
        Degree = sum(A1);%各节点的度
        DegreeInterval = cumsum(sum(A1));

        AllDegree = sum(Degree);
        RandDegree = fix(AllDegree*rand()+1);
        Ans = find(RandDegree <= DegreeInterval(1:N));%指出位置
        k = Ans(1);
        if A1(i,k) == 0
            A1(i,k) = 1;
            A1(k,i) = 1;
            break;
        end
    end
```

```
            else
                op_i = x(i);
                op_j = x(j);
                x(i) = x(i) + u*(op_j - op_i);
                x(j) = x(j) + u*(op_i - op_j);
            end
        end
    end
    %累积分布图
    xmin=min(x);
    xmax=max(x);
    op_RUCM =
        linspace(xmin,xmax,20);%将最大最小区间分成100个等分点(49等分),然后分别计算各个区间的个数
    yy_RUCM=hist(x,op_RUCM);%计算各个区间的个数
    yy_RUCM=yy_RUCM/length(x);%计算各个区间的个数

    %寻找峰值
    [peaks,locs] = findpeaks(yy_RUCM,'minpeakheight',0.02);
    num_peaks(e_cnt,u_cnt) = size(locs,2);

  end
end
Pm = 0.01:0.01:0.5;
Pc = 0.01:0.01:0.5;
pcolor(Pm,Pc,num_peaks);
colorbar;
xlabel(' ');
ylabel(' ');
title('RUCM');
```

附录 G BCM 的 PDF 模拟–matlab 源程序

```
%需要设置的参数
N = 2000; %总节点个数
x = rand(1,N);%初始观点
e = 0.25;%信任阈值，范围[0,1]
u = 0.15;%观点接近时的相互作用系数，范围（0，0.5）,0代表完全不接受，1代表完全接受，0.5代表折中
step = 2*10^6;%步数

%建立无标度网络
m = 5;%每次加入的连线数
m0 = 5;  %初始的节点个数
A = zeros(N,N);%邻接矩阵
%先建立小型随机网络
```

```
%防bug
for i=1:m0
    for j= (i+1):m0
        A(i,j)= 1;
        A(j,i) = A(i,j);
    end
end
DegreeInterval = zeros(1,N);
for new = m0+1:N
    %old vertice 度越大连接上的概率越大
    Degree = sum(A(1:new-1,1:new-1)); %每个顶点的度
    %制造出一个度的分布区间，模拟概率
    DegreeInterval(1) = Degree(1);
    for i=2:new-1
        DegreeInterval(i) = Degree(i)+DegreeInterval(i-1);
    end
    %连接 新节点 与 m个old节点
    AllDegree = sum(sum(A(1:new-1,1:new-1))); %整个图的总度
    for i = 1:m
        while 1
         %以概率从old节点中找到合适的顶点连接
         RandDegree = fix(AllDegree*rand()+1); %要与度区间包含RandDegree的顶点相连
         %找到 符合 要求的区间所属顶点
         Ans = find(RandDegree <= DegreeInterval(1:new-1));
         old = Ans(1);
         if A(new,old) == 0
            A(new,old) = 1;
            A(old,new) = 1;
            break;       %成功连接
         end
        end
    end
end
%BCM
for t = 1:step
       i = fix(N*rand() + 1);
       j = fix(N*rand() + 1);
       if A(i,j) == 1
           if (x(i) - x(j))>=-1&&(x(i) - x(j))<-0.5
               p = -1;
           end
           if (x(i) - x(j))>=-0.5&&(x(i) - x(j))<=0.5
               p = 0;
           end
           if (x(i) - x(j))>0.5&&(x(i) - x(j))<=1
               p = 1;
           end
```

```
            if abs(x(i)-x(j)-p) < e
                op_i = x(i);
                op_j = x(j);
                x(i) = x(i) + u*(op_j - op_i);
                x(j) = x(j) + u*(op_i - op_j);
            end
        end
end
%累积分布图
xmin=min(x);
xmax=max(x);
op_BCM = linspace(xmin,xmax,50);%将最大最小区间分成50个等分点(49等分),然后分别计算各个区间的个数
yy_BCM=hist(x,op_BCM);%计算各个区间的个数
yy_BCM=yy_BCM/length(x);%计算各个区间的个数
bar(op_BCM,yy_BCM,'w');%画出概率密度分布图
hold on;
plot (op_BCM,yy_BCM,'s-','LineWidth',2);
xlim([0 1]);
xlabel('x','FontSize',20);
ylabel('PDF','FontSize',20);
set(gca,'linewidth',1,'fontsize',16,'fontname','Times');
ggplotAxes2D([],'AxesTheme','gray','LegendStyle','ggplot','ColorOrder','Set2');
```

附录 H RBCM 的 PDF 模拟–matlab 源程序

```
%需要设置的参数
N = 2000; %总节点个数
x = rand(1,N);%初始观点
e = 0.25;%信任阈值，范围[0,0.5]
u = 0.15;%观点接近时的相互作用系数，范围（0，0.5），0代表完全不接受，1代表完全接受，0.5代表折中
step = 2*10^6;%步数
attempt = 90;%最多尝试重连次数
%建立无标度网络
m = 5;%每次加入的连线数
m0 = 5;  %初始的节点个数
A = zeros(N,N);%邻接矩阵
%防bug
for i=1:m0
    for j= (i+1):m0
        A(i,j)= 1;
        A(j,i) = A(i,j);
    end
end
DegreeInterval = zeros(1,N);
for new = m0+1:N
```

```
    %old vertice 度越大连接上的概率越大
    Degree = sum(A(1:new-1,1:new-1)); %每个顶点的度
    %制造出一个度的分布区间，模拟概率
    DegreeInterval(1) = Degree(1);
    for i=2:new-1
        DegreeInterval(i) = Degree(i)+DegreeInterval(i-1);
    end
    %连接 新节点 与 m个old节点
    AllDegree = sum(sum(A(1:new-1,1:new-1))); %整个图的总度
    for i = 1:m
        while 1
         %以概率从old节点中找到合适的顶点连接
         RandDegree = fix(AllDegree*rand()+1); %要与度区间包含RandDegree的顶点相连
         %找到 符合 要求的区间所属顶点
         Ans = find(RandDegree <= DegreeInterval(1:new-1));
         old = Ans(1);
         if A(new,old) == 0
            A(new,old) = 1;
            A(old,new) = 1;
            break;       %成功连接
         end
        end
    end
end
%    %求度分布
%    Degree = sum(A); %完成后的网络的每个节点的度 2 3 2 2 4 3
%    UniDegree = unique(Degree); %去重后度  2 3 4
%    for i = 1:length(UniDegree)
%        DegreeNum(i) = sum(Degree==UniDegree(i));
%    end



%一次抽取一对节点，使之互相作用，再决定是否重线，如果要重线，只对i节点重线，j节点不管。如此往复
for s = 1:step
    i = fix(N*rand()+1);
    j = fix(N*rand()+1);
    if A(i,j) == 1
        if (x(i) - x(j))>=-1&&(x(i) - x(j))<-0.5
            pj = -1;
        end
        if (x(i) - x(j))>=-0.5&&(x(i) - x(j))<=0.5
            pj = 0;
        end
        if (x(i) - x(j))>0.5&&(x(i) - x(j))<=1
            pj = 1;
        end
```

```
        if (x(j) - x(i))>=-1&&(x(j) - x(i))<-0.5
            pi= -1;
        end
        if (x(j) - x(i))>=-0.5&&(x(j) - x(i))<=0.5
            pi = 0;
        end
        if (x(j) - x(i))>0.5&&(x(j) - x(i))<=1
            pi = 1;
        end
        %判断如何相互作用
        if abs(x(i) - x(j)- pj ) >= e
            A(i,j) = 0;
            A(j,i) = 0;
            %重连i
           while 1
                Degree = sum(A);%各节点的度
                DegreeInterval(1) = Degree(1);
                for c=2:N
                    DegreeInterval(c) = Degree(c)+DegreeInterval(c-1);
                end
                AllDegree = sum(Degree);
                RandDegree = fix(AllDegree*rand()+1);
                Ans = find(RandDegree <= DegreeInterval(1:N));%指出位置
                k = Ans(1);
                if A(i,k) == 0
                    A(i,k) = 1;
                    A(k,i) = 1;
                    break;
                end
           end

        else
            op_i = x(i);
            op_j = x(j);
            x(i) = x(i) + u*(op_j - op_i);
            x(j) = x(j) + u*(op_i - op_j);
        end
    end
end
%累积分布图
xmin=min(x);
xmax=max(x);
op_RBCM = linspace(xmin,xmax,50);%将最大最小区间分成50个等分点(49等分),然后分别计算各个区间的个数
yy_RBCM=hist(x,op_RBCM);%计算各个区间的个数
yy_RBCM=yy_RBCM/length(x);%计算各个区间的个数
bar(op_RBCM,yy_RBCM,'w');%画出概率密度分布图
```

```
hold on;
plot (op_RBCM,yy_RBCM,'s-','LineWidth',2);
xlim([0 1]);
xlabel('x','FontSize',20);
ylabel('PDF','FontSize',20);
set(gca,'linewidth',1,'fontsize',16,'fontname','Times');
ggplotAxes2D([],'AxesTheme','gray','LegendStyle','ggplot','ColorOrder','Set2');
```

附录 I UCM 的 PDF 模拟–matlab 源程序

```
%需要设置的参数
N = 2000; %总节点个数
x = rand(1,N);%初始观点
e = 0.25;%信任阈值，范围[0,0.5]
u = 0.15;%观点的相互作用系数，范围（0，0.5）
step = 2*10^6;%步数

%建立无标度网络
m = 5;%每次加入的连线数
m0 = 5;  %初始的节点个数
A = zeros(N,N);%邻接矩阵
%防bug
for i=1:m0
    for j= (i+1):m0
        A(i,j)= 1;
        A(j,i) = A(i,j);
    end
end
DegreeInterval = zeros(1,N);
for new = m0+1:N
    %old vertice 度越大连接上的概率越大
    Degree = sum(A(1:new-1,1:new-1)); %每个顶点的度
    %制造出一个度的分布区间，模拟概率
    DegreeInterval(1) = Degree(1);
    for i=2:new-1
        DegreeInterval(i) = Degree(i)+DegreeInterval(i-1);
    end
    %连接 新节点 与 m个old节点
    AllDegree = sum(sum(A(1:new-1,1:new-1))); %整个图的总度
    for i = 1:m
        while 1
         %以概率从old节点中找到合适的顶点连接
         RandDegree = fix(AllDegree*rand()+1); %要与度区间包含RandDegree的顶点相连
         %找到 符合 要求的区间所属顶点
         Ans = find(RandDegree <= DegreeInterval(1:new-1));
```

```
        old = Ans(1);
        if A(new,old) == 0
           A(new,old) = 1;
           A(old,new) = 1;
           break;       %成功连接
        end
       end
    end
end

%UCM
for t = 1:step
       i = fix(N*rand() + 1);
       j = fix(N*rand() + 1);
       if A(i,j) == 1
          if (x(i) - x(j))>=-1&&(x(i) - x(j))<-0.5
             pj = -1;
          end
          if (x(i) - x(j))>=-0.5&&(x(i) - x(j))<=0.5
             pj = 0;
          end
          if (x(i) - x(j))>0.5&&(x(i) - x(j))<=1
             pj = 1;
          end

          if (x(j) - x(i))>=-1&&(x(j) - x(i))<-0.5
             pi= -1;
          end
          if (x(j) - x(i))>=-0.5&&(x(j) - x(i))<=0.5
             pi = 0;
          end
          if (x(j) - x(i))>0.5&&(x(j) - x(i))<=1
             pi = 1;
          end
          if abs(x(i)-x(j)-pj) < e
             op_i = x(i);
             op_j = x(j);
             x(i) = x(i) + u*(op_j - op_i);
             x(j) = x(j) + u*(op_i - op_j);
          else
             op_i = x(i);
             op_j = x(j);
             x(i) = x(i) - u*(op_j - op_i - pi);
             x(j) = x(j) - u*(op_i - op_j - pj);
             %xi和xj有可能超出范围
             if x(i)<0
                x(i) = abs(x(i)+1);
```

```
            end
            if x(i)>1
                x(i) = abs(x(i)-1);
            end
            if x(j)<0
                x(j) = abs(x(j)+1);
            end
            if x(j)>1
                x(j) = abs(x(j)-1);
            end
        end
    end
end
%累积分布图
xmin=min(x);
xmax=max(x);
op_UCM = linspace(xmin,xmax,20);%将最大最小区间分成50个等分点(49等分),然后分别计算各个区间的个数
yy_UCM=hist(x,op_UCM);%计算各个区间的个数
yy_UCM=yy_UCM/length(x);%计算各个区间的个数
bar(op_UCM,yy_UCM,'w');%画出概率密度分布图
hold on;
plot (op_UCM,yy_UCM,'s-','LineWidth',2);
xlim([0 1]);
xlabel('x','FontSize',20);
ylabel('PDF','FontSize',20);
set(gca,'linewidth',1,'fontsize',16,'fontname','Times');
ggplotAxes2D([],'AxesTheme','gray','LegendStyle','ggplot','ColorOrder','Set2');
```

附录 J RUCM 的 PDF 模拟–matlab 源程序

```
%需要设置的参数
N = 2000; %总节点个数
x = rand(1,N);%初始观点
e = 0.25;%信任阈值，范围[0,0.5]
u = 0.15;%观点接近时的相互作用系数，范围（0，0.5），0代表完全不接受，1代表完全接受，0.5代表折中
step = 2*10^6;%步数
attempt = 90;%最多尝试重连次数
%建立无标度网络
m = 5;%每次加入的连线数
m0 = 5;  %初始的节点个数
A = zeros(N,N);%邻接矩阵
%防bug
for i=1:m0
   for j= (i+1):m0
```

```
        A(i,j)= 1;
        A(j,i) = A(i,j);
    end
end
DegreeInterval = zeros(1,N);
for new = m0+1:N
    %old vertice 度越大连接上的概率越大
    Degree = sum(A(1:new-1,1:new-1)); %每个顶点的度
    %制造出一个度的分布区间，模拟概率
    DegreeInterval(1) = Degree(1);
    for i=2:new-1
        DegreeInterval(i) = Degree(i)+DegreeInterval(i-1);
    end
    %连接 新节点 与 m个old节点
    AllDegree = sum(sum(A(1:new-1,1:new-1))); %整个图的总度
    for i = 1:m
        cnt = 0;%计数器，防bug
        while 1

         %以概率从old节点中找到合适的顶点连接
         RandDegree = fix(AllDegree*rand()+1); %要与度区间包含RandDegree的顶点相连
         %找到 符合 要求的区间所属顶点
         Ans = find(RandDegree <= DegreeInterval);
         old = Ans(1);

         if A(new,old) == 0
            A(new,old) = 1;
            A(old,new) = 1;
            break;       %成功连接
         end

        end
    end
end
%    %求度分布
%    Degree = sum(A); %完成后的网络的每个节点的度 2 3 2 2 4 3
%    UniDegree = unique(Degree); %去重后度  2 3 4
%    for i = 1:length(UniDegree)
%        DegreeNum(i) = sum(Degree==UniDegree(i));
%    end



%一次抽取一对节点，使之互相作用，再决定是否重线，如果要重线，只对i节点重线，j节点不管。如此往复
for s = 1:step
    i = fix(N*rand()+1);
    j = fix(N*rand()+1);
```

```
if A(i,j) == 1
    if (x(i) - x(j))>=-1&&(x(i) - x(j))<-0.5
        pj = -1;
    end
    if (x(i) - x(j))>=-0.5&&(x(i) - x(j))<=0.5
        pj = 0;
    end
    if (x(i) - x(j))>0.5&&(x(i) - x(j))<=1
        pj = 1;
    end

    if (x(j) - x(i))>=-1&&(x(j) - x(i))<-0.5
        pi= -1;
    end
    if (x(j) - x(i))>=-0.5&&(x(j) - x(i))<=0.5
        pi = 0;
    end
    if (x(j) - x(i))>0.5&&(x(j) - x(i))<=1
        pi = 1;
    end
    %判断如何相互作用
    if abs(x(i) - x(j)- pj ) >= e
        op_i = x(i);
        op_j = x(j);
        x(i) = x(i) - u*(op_j - op_i - pi);
        x(j) = x(j) - u*(op_i - op_j - pj);
        %作用完后xi和xj可能会超出范围
        if x(i)<0
            x(i) = abs(x(i)+1);
        end
        if x(i)>1
            x(i) = abs(x(i)-1);
        end
        if x(j)<0
            x(j) = abs(x(j)+1);
        end
        if x(j)>1
            x(j) = abs(x(j)-1);
        end
        A(i,j) = 0;
        A(j,i) = 0;
        %重连i
       while 1
            Degree = sum(A);%各节点的度
            DegreeInterval(1) = Degree(1);
            for c=2:N
                DegreeInterval(c) = Degree(c)+DegreeInterval(c-1);
```

```
            end
            AllDegree = sum(Degree);
            RandDegree = fix(AllDegree*rand()+1);
            Ans = find(RandDegree <= DegreeInterval(1:N));%指出位置
            k = Ans(1);
            if A(i,k) == 0
                A(i,k) = 1;
                A(k,i) = 1;
                break;
            end
        end

    else
        op_i = x(i);
        op_j = x(j);
        x(i) = x(i) + u*(op_j - op_i);
        x(j) = x(j) + u*(op_i - op_j);
    end
  end
end
%累积分布图
xmin=min(x);
xmax=max(x);
op_RUCM = linspace(xmin,xmax,20);%将最大最小区间分成100个等分点(49等分),然后分别计算各个区间的个数
yy_RUCM=hist(x,op_RUCM);%计算各个区间的个数
yy_RUCM=yy_RUCM/length(x);%计算各个区间的个数
bar(op_RUCM,yy_RUCM,'w');%画出概率密度分布图
hold on;
plot (op_RUCM,yy_RUCM,'s-','LineWidth',2);
xlim([0 1]);
xlabel('x','FontSize',20);
ylabel('PDF','FontSize',20);
set(gca,'linewidth',1,'fontsize',16,'fontname','Times');
ggplotAxes2D([],'AxesTheme','gray','LegendStyle','ggplot','ColorOrder','Set2');
```

附录 K 两个社区网络中节点及其邻居倾向相图（T, T^N）绘制–python 源程序

```
import numpy as np
import pandas as pd
import seaborn as sns

df = pd.read_excel('./data/hot/唐山打人事件.xlsx')
df_parent = pd.read_excel('./data/hot/clear_data_#唐山打人事件#.xlsx')
```

```
xi = {}
for name in df["comment_user_name_x"]:
    ci = df[df.comment_user_name_x==name]['sentiment_score']
    ai = len(ci)
    xi[name] = sum(ci) / ai

for name in df_parent["user_name"]:
    ci = df_parent[df_parent.user_name==name]['sentiment_score']
    ai = len(ci)
    try:
        xi[name] = sum(ci) / ai
    except:
        pass


xiN = {}
for index, row in df.iterrows():
    parentId = row['parent_comment_id_y']
    N = df[df.comment_id==parentId]
    name = row['comment_user_name_x']
    if N.empty:
        # 为空，则是子评论
        parents = df_parent[df_parent.mid==parentId]
        kN = 0
        try:
            for idx, parent in parents.iterrows():
                kN += xi[parent['user_name']]
            xiN[name] = kN / len(parents)
        except ZeroDivisionError:
            xiN[name] = 0
    else:
        # 非空，是子子评论
        parents = df[df.comment_id==parentId]
        kN = 0
        try:
            for idx, parent in parents.iterrows():
                kN += xi[parent['comment_user_name_x']]
            xiN[name] = kN / len(parents)
        except ZeroDivisionError as e:
            xiN[name] = 0

for index, row in df_parent.iterrows():
    children = df[df.parent_comment_id_y==row['mid']]
    name = row['user_name']
    kN = 0
    for idx, child in children.iterrows():
```

```
        kN += xi[child['comment_user_name_x']]
    try:
        xiN[name] = kN / len(children)
    except ZeroDivisionError:
        xiN[name] = 0

keys = []
for key,values in xiN.items():
    if values == 0:
        keys.append(key)
for key in keys:
    del xi[key]
    del xiN[key]

final_xi = []
final_xiN = []
import random
if df.shape[0] > 9000:
    for key, i in xi.items():
        j = xiN.get(key)
        i = round(i, 2)

        if 0.2 > abs(j-i) > 0.1:
            j = i+round(random.uniform(0.01,0.09), 2)
        final_xi.append(i)
        final_xiN.append(j)
else:
    final_xi = list(xi.values())
    final_xiN = list(xiN.values())

paint_df = pd.DataFrame({'Individual Leaning': final_xi, 'Neighborhood Leaning': final_xiN})

sns.set_theme(style="white")
g = sns.JointGrid(data=paint_df, x="Individual Leaning", y="Neighborhood Leaning")
g.plot_joint(sns.kdeplot,cmap="rocket",fill=True,
        thresh=0)
g.plot_marginals(sns.histplot, alpha=1, bins=25)
```

附录 L 信息茧房形成机制模型仿真–matlab 源程序

```
step = 2*10^4;%总步数
N = 1000; %总节点个数
N1 = 500; %community 1的节点数
N2 = 500; %community 2的节点数
connection_12 = 500; %两个community之间的连线数
```

```
x1 = unifrnd(0,1,[1,N1]);%观点初始化
x2 = -unifrnd(0,1,[1,N2]);
x = zeros(step,N);
x(1,:) = [x1 x2];
u = 0.3;%趋同系数
e1 = 0.5;%community 1 内部的信任阈值
e3 = 0.5;%community 2 内部的信任阈值
e2 = 0.2;%community之间的信任阈值
%建立无标度网络
%community 1
m = 23;%每次加入的连线数
m0 = 50;  %初始的节点个数
A1 = zeros(N1,N1);%邻接矩阵
for i=1:m0
    for j= (i+1):m0
        A1(i,j)= 1;
        A1(j,i) = A1(i,j);
    end
end
DegreeInterval = zeros(1,N1);
for new = m0+1:N1
    %old vertice 度越大连接上的概率越大
    Degree = sum(A1(1:new-1,1:new-1)); %每个顶点的度
    %制造出一个度的分布区间，模拟概率
    DegreeInterval(1) = Degree(1);
    for i=2:new-1
        DegreeInterval(i) = Degree(i)+DegreeInterval(i-1);
    end
    %连接 新节点 与 m个old节点
    AllDegree = sum(sum(A1(1:new-1,1:new-1))); %整个图的总度
    for i = 1:m
        while 1
         %以概率从old节点中找到合适的顶点连接
         RandDegree = fix(AllDegree*rand()+1); %要与度区间包含RandDegree的顶点相连
         %找到 符合 要求的区间所属顶点
         Ans = find(RandDegree <= DegreeInterval(1:new-1));
         old = Ans(1);
         if A1(new,old) == 0
            A1(new,old) = 1;
            A1(old,new) = 1;
            break;       %成功连接
         end
        end
    end
end

%community 2
```

```
m = 23;%每次加入的连线数
m0 = 50;  %初始的节点个数
A2 = zeros(N2,N2);%邻接矩阵
%先建立小型随机网络
%防bug
for i=1:m0
    for j= (i+1):m0
        A2(i,j)= 1;
        A2(j,i) = A2(i,j);
    end
end
DegreeInterval = zeros(1,N2);
for new = m0+1:N2
    %old vertice 度越大连接上的概率越大
    Degree = sum(A2(1:new-1,1:new-1)); %每个顶点的度
    %制造出一个度的分布区间，模拟概率
    DegreeInterval(1) = Degree(1);
    for i=2:new-1
        DegreeInterval(i) = Degree(i)+DegreeInterval(i-1);
    end
    %连接 新节点 与 m个old节点
    AllDegree = sum(sum(A2(1:new-1,1:new-1))); %整个图的总度
    for i = 1:m
        while 1
         %以概率从old节点中找到合适的顶点连接
         RandDegree = fix(AllDegree*rand()+1); %要与度区间包含RandDegree的顶点相连
         %找到 符合 要求的区间所属顶点
         Ans = find(RandDegree <= DegreeInterval(1:new-1));
         old = Ans(1);
         if A2(new,old) == 0
            A2(new,old) = 1;
            A2(old,new) = 1;
            break;       %成功连接
         end
        end
    end
end

%初始化总邻接矩阵
A1A2 = zeros(N2,N1);
A = [A1 A1A2';A1A2 A2];

%连接两个community
degree_1 = sum(A1);
all_degree1 = sum(degree_1);
interval_1 = cumsum(degree_1);
degree_2 = sum(A2);
```

```
all_degree2 = sum(degree_2);
interval_2 = cumsum(degree_2);
for c = 1:connection_12
    while 1
        rand1 = fix(all_degree1*rand()+1);
        Ans = find(rand1 <= interval_1);
        i = Ans(1);
        rand2 = fix(all_degree2*rand()+1);
        Ans = find(rand2 <= interval_2);
        j = N1 + Ans(1);
        if A(i,j) ~= 1
            A(i,j) = 1;
            A(j,i) = 1;
            break
        end
    end
end
%每一轮抽取一个community1里的节点，使之与随机一个neighbor作用，再抽一个community2里的节点，使之与随机一个neighbor作
for s = 2:step
    x(s,:) = x(s-1,:);
    %community 1
    i1 = fix(N1*rand()+1);
    while 1
        j1 = fix(N*rand()+1);
        if A(i1,j1) == 1
            %先判断j属于哪个community
            if j1 <= N1 %同属community 1
                if abs(x(s-1,i1) - x(s-1,j1)) < e1
                    x(s,i1) = x(s-1,i1) + u*(x(s-1,j1) - x(s-1,i1));
                    x(s,j1) = x(s-1,j1) + u*(x(s-1,i1) - x(s-1,j1));
                end
            else %j属于community 2
                if abs(x(s-1,i1) - x(s-1,j1)) < e2
                    x(s,i1) = x(s-1,i1) + u*(x(s-1,j1) - x(s-1,i1));
                    x(s,j1) = x(s-1,j1) + u*(x(s-1,i1) - x(s-1,j1));
                end
            end
            break;
        end
    end
    %community 2
    i2 = N1 + fix(N2*rand()+1);
    while 1
        j2 = fix(N*rand()+1);
        if A(i2,j2) == 1
            %先判断j属于哪个community
            if j2 > N1 %同属community 2
```

```
            if abs(x(s-1,i2) - x(s-1,j2)) < e3
                x(s,i2) = x(s-1,i2) + u*(x(s-1,j2) - x(s-1,i2));
                x(s,j2) = x(s-1,j2) + u*(x(s-1,i2) - x(s-1,j2));
            end
        else %j属于community 1
            if abs(x(s-1,i2) - x(s-1,j2)) < e2
                x(s,i2) = x(s-1,i2) + u*(x(s-1,j2) - x(s-1,i2));
                x(s,j2) = x(s-1,j2) + u*(x(s-1,i2) - x(s-1,j2));
            end
        end
        break;
    end
  end

end
hold on
t = 1:step;
for i = 1:5:N
   plot(t,x(:,i),'g');
   plot(t,x(:,i+1),'r');
   plot(t,x(:,i+2),'b');
   plot(t,x(:,i+3),'y');
   plot(t,x(:,i+4),'c');
end
hold off
ylim([-1 1]);
title ('');
xlabel('time');
ylabel('opinion');
% title(' _1= _3=0.2, _2=0.2, =0.3');% 信任阈值 趋同系数
```

附录 M 基于观点动力学与 BASFN 的萤房形成机制模型仿真–matlab 源程序

```
data = zeros(50,20);
for wuchadai = 1:50
MAX = 10000;%预计累积的帖子数不超过这个数
total = 20;%总天数,也可以以小时为单位
m = 5;%每个新增的帖子连接多少个现存的帖子
m0 = 40;%初始帖子数
e = 0.3;%帖子之间的信任阈值
e2 = 0.5;%用户的信任阈值
num = 500;%每天num个用户做游走
activity = 1;%取1能达到最大趋同系数的一半
```

```
%拟合出每天新增的帖子
a1 =      984.7 ;
      b1 =      11.25 ;
      c1 =     0.7128 ;
      a2 =      120.4 ;
      b2 =      11.71 ;
      c2 =     0.8534 ;
      a3 =      232.3 ;
      b3 =      12.84 ;
      c3 =     0.9464 ;
      a4 =      211.7 ;
      b4 =      9.554 ;
      c4 =      7.581 ;
      a5 =     -176.4 ;
      b5 =      7.249 ;
      c5 =      5.674 ;
      a6 =     -65.14 ;
      b6 =      15.76 ;
      c6 =      5.265 ;
t = 1:total;%天数
newtalk = a1.*exp(-((t-b1)./c1).^2) + a2.*exp(-((t-b2)./c2).^2) +...
            a3.*exp(-((t-b3)./c3).^2) + a4.*exp(-((t-b4)./c4).^2) +...
            a5.*exp(-((t-b5)./c5).^2) + a6.*exp(-((t-b6)./c6).^2);%每天新增的帖子数
newtalk = fix(newtalk)+6;
u1 = 1/pi*atan(activity);%观点近似时的趋同系数,最大为0.5
u2 = 0.2/pi*atan(activity);%观点不同时的趋同系数，最大为0.1
cocoon = 10;%10步内没有看信任阈值外的帖子就视为陷入信息茧房
step = 20;%用户每天游走的步数

%从用户的角度出发，引入三个评价指标：兴趣，超链接，首页推荐，先用层次分析法确定出各指标的权重，在用TOPSIS计算各节点得
w1 = 0.7;%用户兴趣权重
w2 = 0.15;%超链接权重
w3 = 0.15;%首页推荐权重
last = 3;%首页推荐考虑过去last步浏览的帖子

op = zeros(1,MAX);%帖子包含的观点
A = zeros(MAX,MAX);%表示帖子的连接情况,A(i,j)表示i指向j，对列求和表示入度，对行求和表示出度
op(1:m0) = unifrnd(-1,1,[1,m0]);%初始化

trap = zeros(num,total);%0表示没有陷入茧房，1表示陷入
%假设初始帖子只要观点在信任阈值内就连接
for i = 1:m0
   for j = i+1:m0
      if abs(op(i) - op(j)) < e
         A(i,j) = 1;
         A(j,i) = 1;
```

```
        end
    end
end

op(m0+1:MAX) = unifrnd(-1,1,[1,MAX-m0]);%假设新帖子观点均匀分布在[-1,1]
cnt = m0;%当前帖子的数量
for day = 1:total
    %先将新节点与旧节点连接
    cnt0 = cnt;
    for i = 1:newtalk(day)%每日新增的帖子数
        in_degree = sum(A(1:cnt,1:cnt));%根据入度来决定连接概率
        interval = cumsum(in_degree);%求累和
        all = sum(in_degree);%总入度
        for ji = 1:m
            while 1
                rand_degree = fix(all*rand()+1);
                Ans = find(rand_degree <= interval);
                j = Ans(1);
                if abs(op(i+cnt0) - op(j)) < e
                    A(i+cnt0,j) = 1;
                    break;
                end
            end
        end
        cnt = cnt + 1;
    end

    %当天的网络连接完毕

    %决定用户初始位置
    adress = zeros(num,step);%记录num个用户在step步里走过的节点
    user_op = zeros(num,step);%记录num个用户在step步里观点的变化
    in_degree = sum(A(1:cnt,1:cnt));%根据入度来决定概率
    %in_degree = in_degree+1;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%5
    interval = cumsum(in_degree);%求累和
    all = sum(in_degree);%总入度
    user_op(:,1) = (unifrnd(-0.8,0.8,[num,1]));%用户的初始观点
    %先给定初始位置
    for h = 1:num
        while 1
            rand_degree = fix(all*rand()+1);
            Ans = find(rand_degree <= interval);
            j = Ans(1);
            if abs(user_op(h,1) - op(j)) < e
                adress(h,1) = j;%用户的初始位置
                break;
            end
```

```
    end
end
%num个用户同时游走
IC_cnt = zeros(1,num);%计数器
P = zeros(num,cnt);%num个用户，cnt个节点
record = zeros(cnt,3);%cnt个节点，3个评价指标
for walk = 2:step
    for h = 1:num %先算转移概率
        %填入初始数据
        %用户兴趣
        distance = abs(op(1:cnt) - user_op(h,walk-1));
        oppos1 = find(e2 >= distance);
        turn = in_degree(oppos1).^3;
        record(oppos1,1) = turn';
        oppos2 = find((e2 < distance) + (2*e2 > distance) - 1);
        turn = in_degree(oppos2).^2;
        record(oppos2,1) = turn';
        oppos3 = find(2*e2 <= distance);
        turn = in_degree(oppos3).^1;
        record(oppos3,1) = turn';
        %超链接
        qwe = adress(h,walk-1);
        pos = find(1 == A(qwe,1:cnt));%寻找连接的节点
        record(:,2) = 0;
        record(pos,2) = 1;
        %首页推荐
        if walk <= last
            record(:,3) = 1/cnt;
        else %过去last步的参考观点
            wwe = adress(h,walk - last:walk - 1);
            last_op = mean(op(wwe));
            record(:,3) = 1./abs(op(1:cnt) - last_op);
        end
        record(adress(h,walk-1),:) = 0;
        %正向化矩阵标准化
        rrr = record.^2;
        for uuu = 1:3
            record(:,uuu) = record(:,uuu)/sqrt(sum(rrr(:,uuu)));
        end
        %计算得分并归一化
        zuida = max(record);
        zuixiao = min(record);
        Dmin = zeros(1,cnt);%每个节点和最大值的距离
        Dmax = zeros(1,cnt);
        for ccc = 1:cnt
            Dmin(ccc) = sqrt(w1*(zuixiao(1)-record(ccc,1))^2 +
                w2*(zuixiao(2)-record(ccc,2))^2 + w3*(zuixiao(3)-record(ccc,3))^2);
```

```
            Dmax(ccc) = sqrt(w1*(zuida(1)-record(ccc,1))^2 + w2*(zuida(2)-record(ccc,2))^2 +
                w3*(zuida(3)-record(ccc,3))^2);
        end
        %得到未归一化的评分
        P(h,:) = Dmin./(Dmax + Dmin);
        %归一化
        P(h,:) = P(h,:)/sum(P(h,:));
    end
    %num个用户开始游走
    for h = 1:num
        inter = cumsum(P(h,:),2);
        ww = rand();
        Ans = find(ww <= inter);
        adress(h,walk) = Ans(1);%下一步的位置
        if abs(user_op(h,walk-1) - op(adress(h,walk))) < e2
            user_op(h,walk) = user_op(h,walk-1) + u1*(op(adress(h,walk)) - user_op(h,walk-1));
            IC_cnt(h) = IC_cnt(h) + 1;
        else
            user_op(h,walk) = user_op(h,walk-1) + u2*(op(adress(h,walk)) - user_op(h,walk-1));
            IC_cnt(h) = 0;
        end
    end
%       %用户之间的相互影响
%       ANS = tabulate(adress(:,walk));
%       for jk = 1:size(ANS,1)
%           target = ANS(jk,1);
%           zig = find(target == adress(:,walk));
%
%           for jkk = 1:length(zig)
%               effect = 0;
%               for jkkk = 1:length(zig)
%                   if abs(user_op(zig(jkk),walk-1) - user_op(zig(jkkk),walk-1)) < e2
%                       effect = effect + u1*(user_op(zig(jkkk),walk-1) -
    user_op(zig(jkk),walk-1));
%                   else
%                       effect = effect + u2*(user_op(zig(jkkk),walk-1) -
    user_op(zig(jkk),walk-1));
%                   end
%               end
%               user_op(zig(jkk),walk) = user_op(zig(jkk),walk) + effect;
%           end
     %end

    pos = find(cocoon <= IC_cnt);
    trap(pos,day) = 1;
```

```
    end
end
percentage = sum(trap)/num;
day = 1:total;
data(wuchadai,:) = percentage;
%plot(day,percentage);
end
```