运用SNA分析交际圈中的中心人物与枢纽人物

——以虚拟主播为例

作者姓名	徐一鸣 曹鼎言 黄炳赫
所在省份、城市	江苏扬州
学校	江苏省扬州中学
年级	高二
指导教师姓名	孔令辉

摘要

当今社会关系纷繁复杂。小组选取了某一时刻的虚拟主播交际圈为样本范围建立模型，采用SNA（社会网络分析）分析交际圈中的中心人物、枢纽人物和边缘人物，并据此绘制出鲜明直观的虚拟主播星系图，在一定程度上量化认识了虚拟主播交际网络。

关键词：SNA社会网络分析；虚拟主播；交际圈；GN算法及其优化；星系

2022年3月

1. 问题分析

随着社会的高速发展，人们的交际圈也在发生着变化。朋友圈、动漫圈、游戏圈……大大小小的社群中存在着形形色色的人，每个人都在社群中占据着一定的地位：有的人是这个社群中的核心（中心人物），有的人则是社群与社群间的桥梁（枢纽人物），当然也有的人仅仅只是社群中的“小透明”（边缘人物）。

在人际交往中，为了更好地融入圈中，了解一个社群中的**中心人物、枢纽人物和边缘人物**就显得尤为重要。理论上，我们使用的SNA社会网络分析可以分析圈中的这种人际关系。但在真实的社会中，影响人际关系的因素十分复杂，尽管随着因素种类的增加，结果会更贴切实际，但也会造成计算量指数级上升。在保证模型的贴合性的同时，我们做出适当的简化，选取一个容易获得数据的交际圈——虚拟主播交际圈为样本和对象，进行建模，得出该社群中的中心人物、枢纽人物和边缘人物，并以此为基础绘制虚拟主播交际圈的星系图（关系图）。

2.模型假设

经过小范围调查我们得知一个人在交际圈中的影响力与以下因素有关：

(I) 内因

- 外貌因素（颜值）

  这是小组成员最先想到的因素。但虚拟主播的形象是人为设计的，本身就迎合了大众的口味，去刻意评价这一点没有太大的意义。在我们选取的样本范围中，我们认为所有虚拟主播的颜值为同一水平，该参数在本例中不会对影响力产生太大的影响。

- 绝对能力值

  作为一个虚拟主播，影响力必然和自己的能力正相关。自身的能力（歌声、舞蹈等）越强，在圈内的影响力也越大。而作为一个B站的UP主，最能反应能力的数据是视频平均播放量。因此，我们假设视频平均播放量代表绝对能力值。

- 相对能力值

  很显然，一个虚拟主播在圈内的影响力最直观的数据就是粉丝的数量。相较粉丝数量，粉丝数量之比更具有研究价值。因此，小组假设（粉丝数量/最高粉丝数量）代表相对能力值。

(II) 外因

- 共同爱好

  在一般的交际圈中这一点尤为重要。但对于虚拟主播这一特殊的交际圈中，这一点和外貌因素同样可以忽略不计。

- 共同朋友

  朋友的交叉程度可以鲜明地体现出两者之间联系是否紧密，以便于我们构建社交网络。鉴于粉丝数量的庞大，我们难以获取粉丝重合度的数据。幸运地是，舰长数量不仅能很好地反应粉丝数量，还有较小的数据量。因此，我们假设粉丝重合度等于舰长重合度。

- 地缘关系

  在互联网中，地缘关系与线下环境不尽相同。我们假设，虚拟主播的地缘关系与平台和其所属公司的企划有关。考虑到我们选取的交际圈范围，是我们身边最具代表性的虚拟主播平台B站，因此我们将地缘关系假设为虚拟主播所属公司的企划。

同时，具体样本的数据值可能随时间存在波动（例如舰长数量、平均播放量等可能稍有增减），我们无法严格获取到同一瞬间的所有状态。但我们认为，在数据爬取的时间尺度下，这种波动属于小量，可以忽略。

3.模型建立

概念定义

在本模型中，我们使用如下定义：

- **定义1.1** 总星系 整个虚拟主播交际圈。
- **定义1.2.1**子星系 整个虚拟主播交际圈中所划分的社群。
- **定义1.2.2** 双星/三星/多星星系 子星系中节点数量为两个/三个/三个以上的星系。
- **定义1.3.1** 中心人物(恒星) 每个子星系中，接近中心性最大的节点。
- **定义1.3.2** 节点的接近中心性 一个节点到所有其他节点的最短路径距离的和的倒数。
- **定义1.4.1** 枢纽人物(跃迁门) 中介中心性较大的节点。
- **定义1.4.2** 节点的中介中心性 网络中所有最短路径中经过该节点的路径的数目占最短路径总数的比例。
- **定义1.5** 边缘人物(星环) 指双星星系和三星星系中所包含的节点。
- **定义2.1** 直播势虚拟主播 半年内曾进行过直播活动的虚拟主播
- **定义2.2** 舰长重合度 设主播 a, b 拥有的舰长集合为 A, B，舰长重合度$\alpha_{a,b}$ 由下式决定

$$\alpha_{a,b} = \frac{|A \cap B|}{|A|} \tag{1}$$

样本和数据来源

样本：B站（bilibili.com）2022年3月7日6时18分时刻粉丝数Top500的直播势虚拟主播（通过 https://vtbs.moe/ 实时数据获取）。

舰长、视频播放量、视频数量等：小组成员使用JavaScript脚本，通过B站官方API接口（ https://api.live.bilibili.com/ ）爬取，最终数据获取时间为2022年3月7日10时左右。

算法

模型采用SNA（Social Network Analysis）社会网络分析[1]。

在模型假设中，我们已经知道了相关的变量，这里用字母表示：

- 主播 i 的绝对能力值

$$K_i (i \in [1, 500] \cap \mathbb{N}) \tag{2}$$

- 主播 i 的相对能力值

$$\omega_i (i \in [1, 500] \cap \mathbb{N}) \tag{3}$$

- 主播 i,j 之间的舰长重合度

$$\alpha_{i,j}(i,j\in[1,500]\cap\mathbb{N},i\neq j) \tag{4}$$

- 主播 i,j 之间的引力值

$$F_{i,j}(i,j\in[1,500]\cap\mathbb{N},i\neq j) \tag{5}$$

显然，i 对于其他节点的影响力（或者关系紧密程度）与上述变量均为正相关。故当我们考虑一个特定的 i 时，我们所想要求的 $F_{i,j}$ 为关于$K_i,\omega_i,K_j,\omega_j,\alpha_{i,j}$的增函数。

注意到，我们的目的是筛选出影响力相对较大的节点，而不是绝对较大的节点。因此我们不关心 $F_{i,j}$ 具体的函数值，而只关心其相对大小。从而，我们仅需要设计出一个正相关的函数就可以正确筛选出所需的数据。鉴于最终目的是绘制出一幅虚拟主播交际圈星系图，我们不妨假设 $F_{i,j}$ 函数具有万有引力定律的形式，即

$$F_{i,j}=G\frac{M_i\times M_j}{r^2}\propto\frac{M_i\times M_j}{r^2}, \tag{6}$$

不妨取 $G=1$，则

$$F_{i,j}=\frac{M_i\times M_j}{r^2}, \tag{7}$$

其中

$$M_i=\sqrt{K_i\times\omega_i}, \tag{8}$$

$$\frac{1}{r}=max\left\{\alpha_{i,j},\alpha_{j,i}\right\}. \tag{9}$$

r 的方向满足

$$\begin{aligned}&\alpha_{i,j}\geq\alpha_{j,i},i\text{指向}j\\&\alpha_{i,j}\leq\alpha_{j,i},j\text{指向}i\end{aligned} \tag{10}$$

当满足以下任意一种情形，我们给节点 i 、j 之间建立有向边：

情形一：i 与 j 同属一个公司下的同一企划；

情形二：i 与 j 组成的 (i,j) 对，满足 $F_{i,j}$ 排序在 $F_{i,x},x\in[1,500]\cap\mathbb{N}$ 的前 $\frac{1}{5}$ 部分；

上述操作完成后，我们便获得了500个点之间的有向图。在有向图的基础上进行SNA社会网络分析便可以得到我们所需要的中心人物、枢纽人物和边缘人物。

4.模型计算

4.1 数据

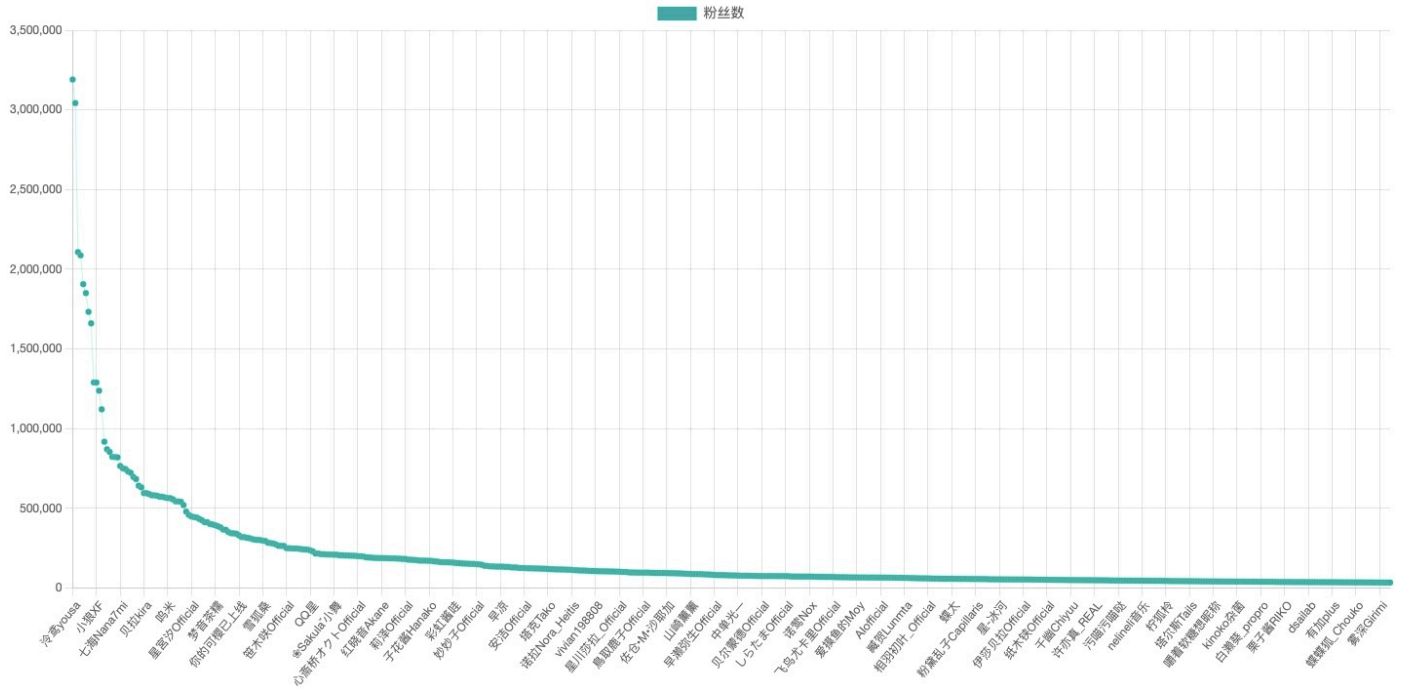

图1.虚拟主播Top500粉丝数

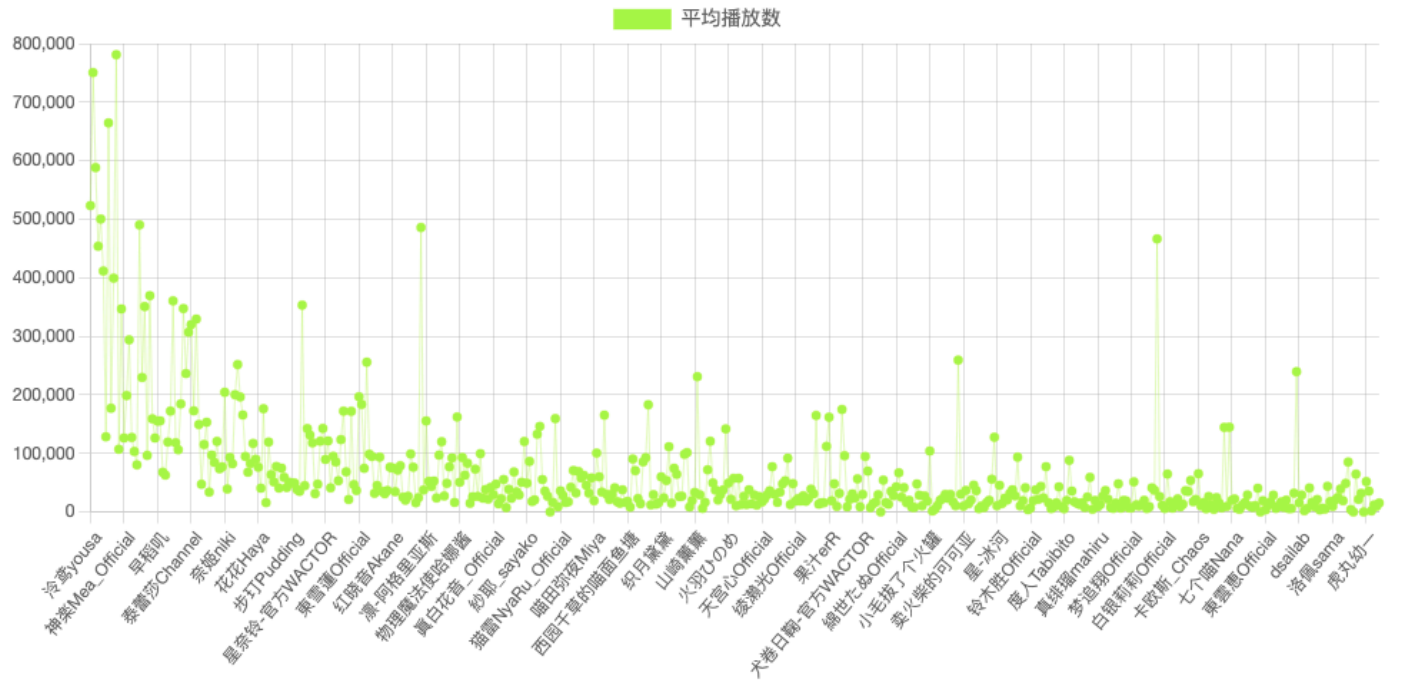

图2.虚拟主播Top500视频平均播放量

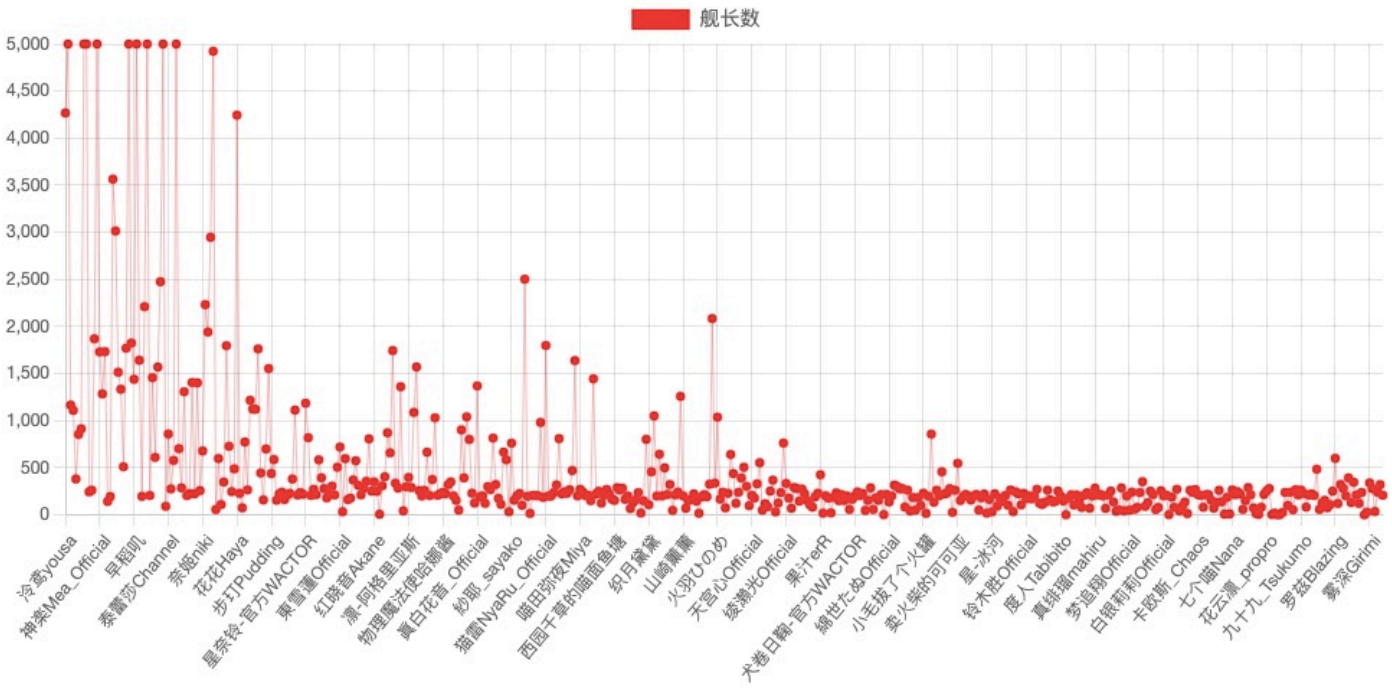

图3.虚拟主播Top500舰长数（封顶5000）

表1.虚拟主播Top500引力值（节选Top12）

	泠鸢yousa	hanser	祖娅纳惜	多多poi丶	神楽七奈Official	鹿乃ちゃん	AIChannel官方	嘉然今天吃什么	C酱です	小狼XF	一块电鹿板	冰糖IO
泠鸢yousa	/	62812.7	33645.58	29121.58	20763.21	18188.8	3761.19	1993.63	1179.15	1186.02	0	8930.93
hanser	62812.7	/	22613.72	29903.83	6070.55	9185.4	977.48	597.17	724.38	1387.03	0	4125.16
祖娅纳惜	33645.58	22613.72	/	4283.83	3577.89	2137	360.07	628.74	715.15	0	0	3318.53
多多poi丶	29121.58	29903.83	4283.83	/	3909.8	1556.83	629.55	577.38	1050.77	893.33	0	5047.36
神楽七奈Official	20763.21	6070.55	3577.89	3909.8	/	21733.98	4167.27	1687.83	0	896.35	0	5011
鹿乃ちゃん	18188.8	9185.4	2137	1556.83	21733.98	/	2564.45	672.07	917.32	0	0	1884.46
AIChannel官方	3761.19	977.48	360.07	629.55	4167.27	2564.45	/	1019.15	231.84	0	0	1176.68
嘉然今天吃什么	1993.63	597.17	628.74	577.38	1687.83	672.07	1019.15	/	157.38	964.11	0	671.08
C酱です	1179.15	724.38	715.15	1050.77	0	917.32	231.84	157.38	/	1315.92	0	360.84
小狼XF	1186.02	1387.03	0	893.33	896.35	0	0	964.11	1315.92	/	3609.27	636.08
一块电鹿板	0	0	0	0	0	0	0	0	0	3609.27	/	0
冰糖IO	8930.93	4125.16	3318.53	5047.36	5011	1884.46	1176.68	671.08	360.84	636.08	0	/

4.2 分析

得到有向图之后，进行SNA社会网络分析。

要想求出枢纽人物、中心人物和边缘人物，必须先给整个交际圈进行合理的社群划分，即划分“子星系”。

由**定义1.4.2**知，节点的中介中心性的计算公式如下：

$$C_i = \sum_{j,k \in \mathbb{N}} \frac{n_{j,k}(i)}{n_{j,k}} \tag{11}$$

其中，$n_{j,k}$ 表示节点 j、k 之间的最短路径的条数；$n_{j,k}(i)$ 表示节点 j、k 之间的最短路径中经过节点i的条数，即节点介数。

类比点的中介中心性，我们有，

定义1.4.3 边的中介中心性 网络中所有最短路径中经过该边的路径的数目占最短路径总数的比例：

$$E_m = \sum_{p,q \in \mathbb{N}} \frac{n_{p,q}(m)}{n_{p,q}} \tag{12}$$

其中，$n_{p,q}$ 表示以 p 、q 之间的最短路径的条数；$n_{p,q}(m)$ 表示边 p、q 之间的最短路径中经过边 m 的条数，即边介数。

如果一条边连接两个社群，那么这两个社群节点之间的最短路径通过该边的次数就会最多，即相应的边的中介中心性最大。如果删除该边，那么两个社群就会分割开，成为两个相对独立的子星系。因此，我们的算法应当不断计算当前网络的最短路径，计算每条边的中介中心性，然后删除中介中心性最大的边。当无边可删时（即每个点都是一个社群时），算法停止。

但是，我们虽然给出了划分社群的方法，但依旧无法确定哪一种划分才是最合理的。为此，我们引入模块度Q[2]。

定义3.1 模块度 Q 一个网络在某种社群划分下与随机网络的差异。

因为随机网络不具备社群结构，故对应的差距越大，说明社群的划分结果越好。

模块度 Q 的计算公式由下式给出：

$$Q = \sum_i (e_{ii} - a_i^2) = \sum_i e_{ii} - \sum_i a_i^2 \tag{13}$$

其中

$$Q \in [-0.5, 1) \tag{14}$$

i 代表的是第 i 个社群，e_{ii} 表示社群 i 的边占原始网络总边数的比例，a_i 表示所有连接了社群 i 中的顶点的边占原始网络总边数的比例。

将模块度 Q 最大的划分输出，我们便可以划分出最合理的社群。这种算法被称为***GN***（***Girvan-Newman***）算法[3]。

GN算法优点通俗易懂，最容易想到，但缺点是时间复杂度高达 $O(m^2n)$（m为边数，n为节点数），难以在短时间内获得我们想要的数据，且容易将极为重要的节点提前孤立出去，导致孤立节点数量偏多。为此，我们创新性地提出优化方法——GN加速保护算法（见附录一）。

加速：在GN算法的基础上，我们不采取每次只删一条边的方法，而改为每删x条边，计算一次边介数。例如，当 $x = 10$时，我们每删10条边计算一次边介数，这样速度可以提升约10倍。相应地，会造成精度下降，但我们可以获得模块度最大时的模糊边界，只要在边界附近重新精确删边便可以获得较为精确的划分（见表2）。同时，因为模块度变化的大致趋势应该相同，我们可以利用这种方法来进行数据的检验。

保护：为了防止孤立节点出现，我们加入限制条件，限制每个节点至少保留一条边。即使一个节点最后的边边介数足够大，我们也不删这条边。这样不仅可以有效防止过多的孤立社群的出现，还避免了重要节点被提前孤立。

表2.GN加速算法不同加速倍数x的对比

x	Q最大时的已删边次数N	$Q[N-1]$	$Q[N]$	$Q[N+1]$
1	37581	349260150	364569130	360253686
10	37579	340289826	348887538	347799154
50	37580	331404588	333562142	332828106
100	37578	333741748	338414030	337192934
1000	37589	292445014	303725654	298550764

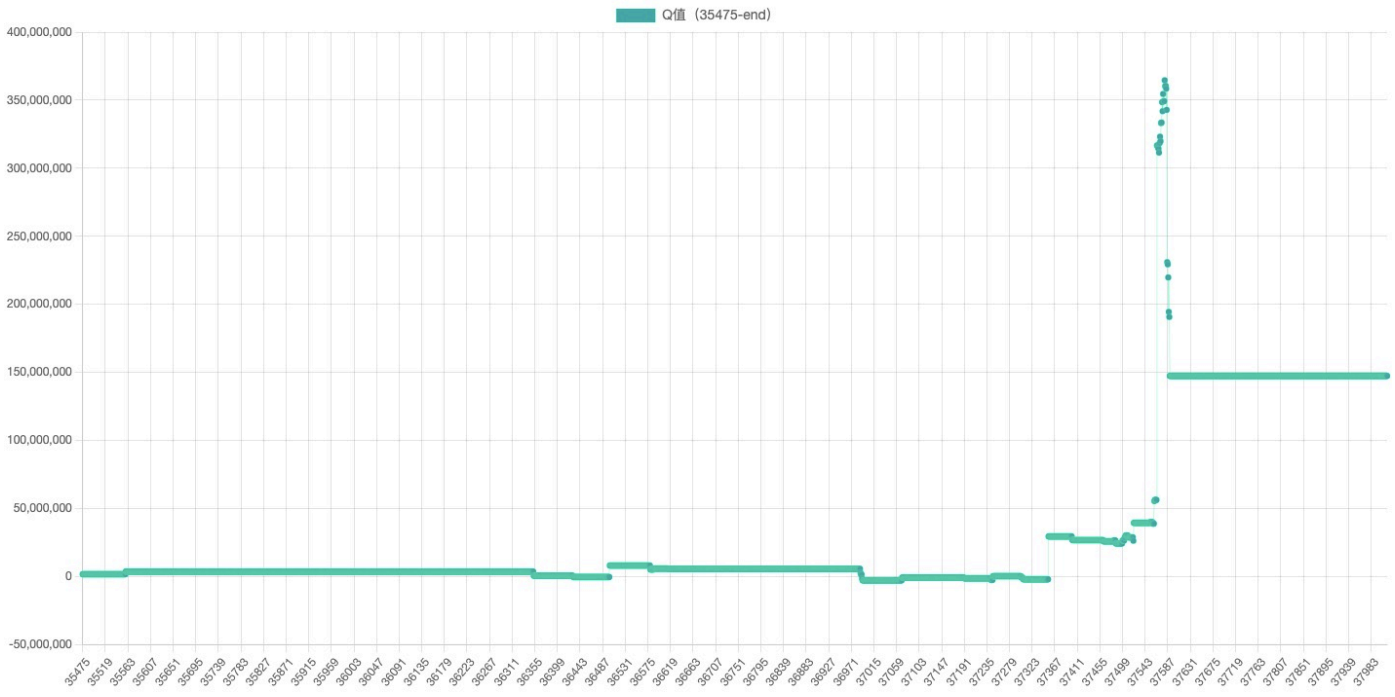

图4.x=1模块度变化散点图（35475-end）

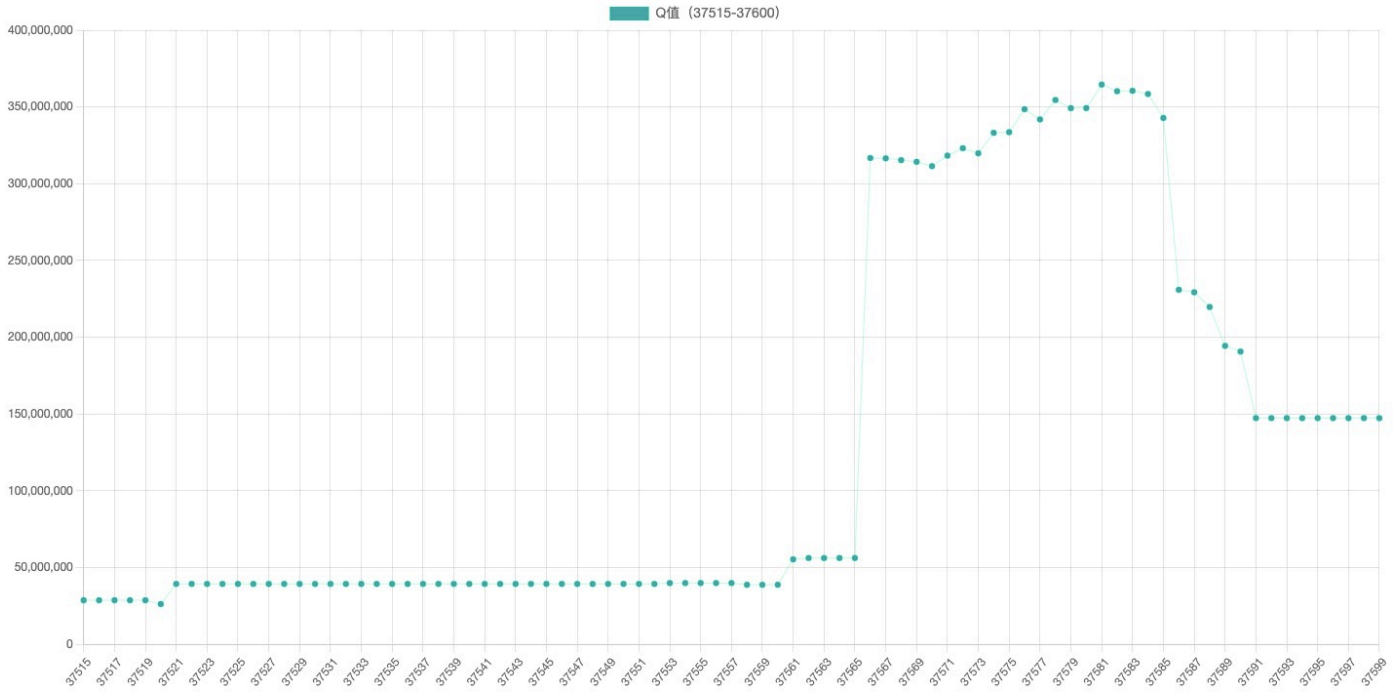

图5.x=1模块度变化散点图（37515-37600）

显然，划分出的社群中可能存在双星星系和三星星系的情况，这些小社群与大社群之间关系并不紧密，他们就是我们所寻找的边缘人物，最后将构成星环。

由定义1.4.1知，所求的枢纽人物就是中介中心性最大的N（N为多星星系的数量）个节点。

将500个点的有向图转化为500×500的邻接矩阵，并应用C++进行编程，（见附录二）求出中介中间性最大的N个节点，即为“跃迁门”（见表3）。

表3.跃迁门一览

编号	所属社群	昵称	中介中心性	编号	所属社群	昵称	中介中心性
5	20	神楽七奈Official	1054.66	113	20	安晴Ankii	1915.88
10	3	小狼XF	858.078	117	20	不举萌栗	913.888
33	20	菜菜子Nanako	3243.67	139	20	筱魈Syou	835.55
48	20	帕里_Paryi	2597.68	149	边缘	龙胆尊Official	860.129
52	20	犬山玉姬Official	2597.68	174	20	まこと-macoto官方	1073.39
60	20	疾風醬	1418.78	179	20	赫卡Tia	2780.14
67	20	琉绮Ruki	2060.55	195	20	田汐汐_Official	1427.09
91	20	QQ星	1183.72	204	20	花留Karu	1164.92
96	20	早见咲Saki	1420.25	232	边缘	魔宫永恋Mamiya	827.165
103	20	時雨羽衣Official	984.888	292	20	露珀塔_Ruperta	2085.43

由**定义1.3.1**知，所求的中心人物就是每个多星星系中接近中心性最大的节点。

由**定义1.3.2**知，接近中心性的计算公式：

$$C(u)=\frac{1}{\sum_{v-1}^{n-1}d(u,v)} \tag{15}$$

其中，u 为待计算接近中心性的节点；n为图中所有节点数；$d(u,v)$ 是节点 u 和节点 v之间的最短距离。

对每一个子星系进行分析，将每个相对独立的多星星系转化为 S 维邻接矩阵（S 为该子星系中的节点数量），并应用C++进行编程（见附录二）。求出每个子星系中接近中心性最大的点，即为“恒星”（见表4）。

表4.恒星一览

编号	所属社群	昵称	接近中心性	编号	所属社群	昵称	接近中心性
62	01	雫るる_Official	15	339	11	卖火柴的可可亚	3
54	02	物述有栖Official	15	295	12	日月咪玉-官方	6
65	03	尼奈nine_Official	3	429	13	团小哈	5
152	04	古堡龙姬	4	83	14	彼岸霜滢Eliro	3
132	05	诗小雅Official	4	373	15	黎歌Neeko_channel	7
317	06	塔克Tako	3	285	16	白雪巴_Official	3
181	07	古守血遊official	4	242	17	茉里Mari	4
13	08	HiiroVTuber	15.8	68	18	可愛的島風	7.5
333	09	唐九夏还想再躺一下	4	408	19	望月希咏	43
364	10	花玲Karin	4	12	20	冰糖IO	147

特别地，总星系的中心人物为编号43的阿梓从小就很可爱，全局接近中心性为71.2857。

4.3结论

在得出各个子星系的图以及所需的枢纽人物、中心人物和边缘人物后，我们得到了相对直观的虚拟主播社会关系网络（图6）。由这张虚拟主播社会关系网络图，可以直观地看出在社群中哪些虚拟主播的影响力较大。

图6.虚拟主播社会关系网络（3D图）

其中，我们可以发现，20个枢纽人物有17个都集中在社群20中，从图中能明显发现社群20还是总星系中最大的子星系。这两点很好地证明了社群20在整个虚拟主播交际圈中的重要性。

另一方面，在枢纽人物和中心人物中不乏我们所熟悉的面孔，但也缺少了一部分我们认为重要的人物，如泠鸢yousa、hanser、嘉然今天吃什么等。经过仔细审视过后发现，缺少这些可以争夺“V圈第一人”的虚拟主播的原因是她们的连边几乎都是入边，而没有出边。这种类似“黑洞”的性质导致难以从她们通往其他的虚拟主播，因而导致中介中心性和接近中心性并不高。

> 附：虚拟主播社会关系网络已上传至小组服务器 http://60.205.178.3:4342/net

5. 模型验证和改进方向

为了验证此模型确实能够正确地给出我们想要的结果，我们手动选取几个小社群分别单独进行SNA社会网络分析。分析得出的结果与前文划分交际圈后得到的社群的结果均相同，且均符合感性认知。这验证了模型确实能给出正确的结果，即枢纽人物和中心人物的正确性。

为了使得到的虚拟主播社会关系网络更加优美，更具有观赏性[4]，我们保持原有的连线不变，根据已知的各个星球的性质（是否为星环、所属的子星系、是否为跃迁门、是否为恒星），以 M_i 代表该虚拟主播的星球大小，r 代表星球之间的距离，通过 **3d-force-graph库**和**threejs库**进行作图，最终得到一张兼具美观性和科学性的虚拟主播星系图（图7）。

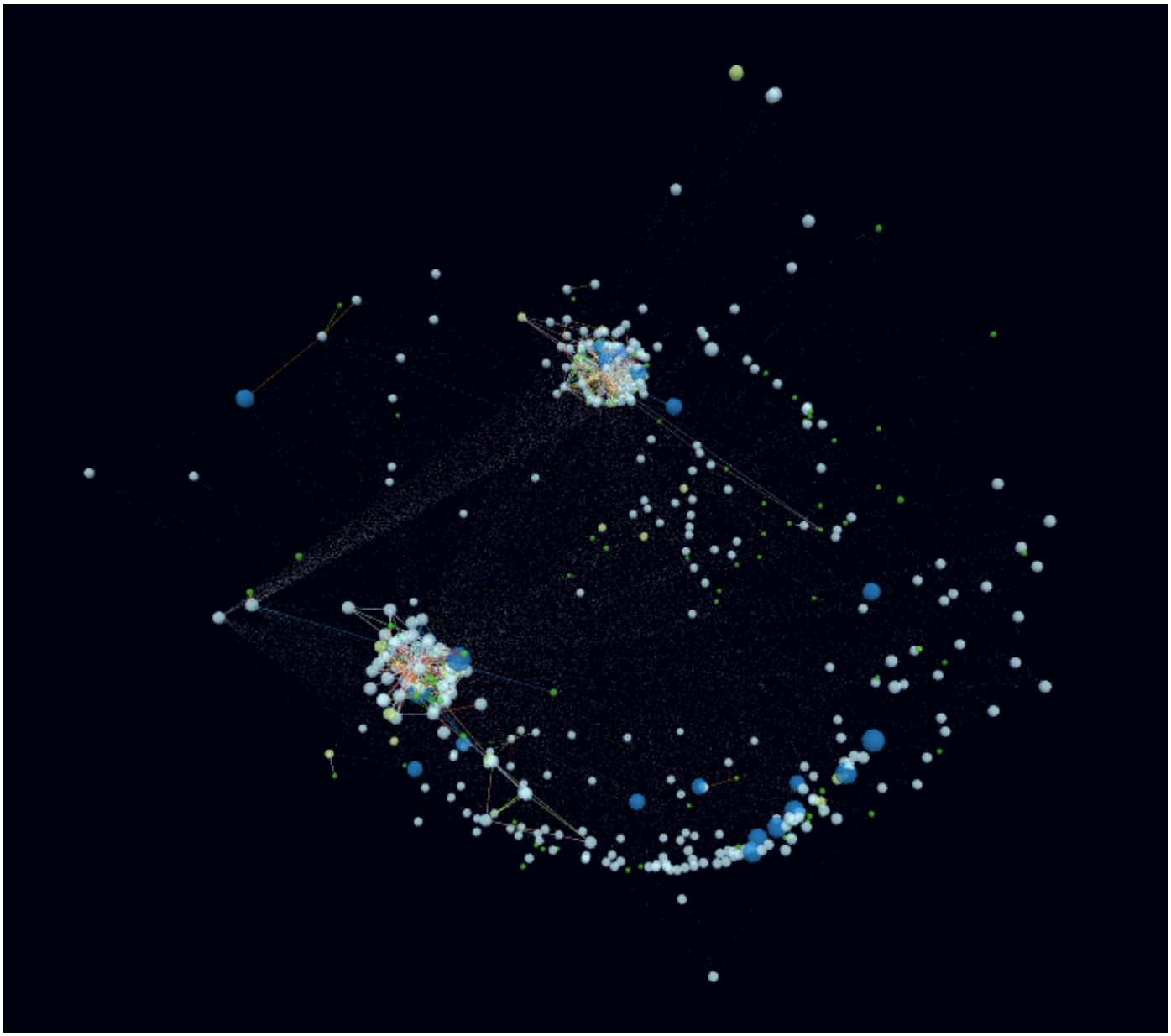

图7.虚拟主播星系图（3D图）

> 附：虚拟主播星系图已上传至小组服务器 http://60.205.178.3:4342/uni

4.3结论中提及的“黑洞”性质将是我们未来准备探究的方向。

此外，我们的模型虽然能较精确地给出交际圈中的社会网络，且具有一定的普适性，但这建立在数据易获取的前提下，在实际生活的交际圈中如何量化主要的影响因素仍有待探究。另一方面，我们选择的划分社群的算法较为基础，虽然在优化过后时间复杂度已经能降低一个数量级，但仍难以完成大型社会网络的社群划分，更高端的 ***FastGreedy*** 算法和 ***BGLL*** 算法将是我们未来的探究方向。

> 附：所有代码、数据已MIT开源至 https://github.com/HomeArchbishop/math-modeling-2022

参考文献

[1]维基百科.Social network analysis[OL].

[2]Newman M E J. Modularity and community structure in networks[J]. Proceedings of the national academy of sciences, 2006, 103(23): 8577-8582.

[3]Girvan M, Newman M E J. Community structure in social and biological networks[J]. Proceedings of the national academy of sciences, 2002, 99(12): 7821-7826.

[4]千鹤穹华_Channel.V圈银河系——璀璨而夺目【b站虚拟主播网络关系图】[OL].2021-07-11

附录

附录一

GN加速保护算法

```
#include <bits/stdc++.h>
#define xh(i,a,b) for(register int (i)=(a);(i)<(b);++(i))
#define ll long long
using namespace std;

template <typename T>
T rd(T& x) {
  x = 0;
  bool f = 1;
  char c = getchar();
  while(c > '9' || c < '0') {
    f = c - '-';
    c = getchar();
  }
  while(c >= '0' && c <= '9') {
    x = x * 10 + c - '0';
    c = getchar();
  }
  x = f ? x : -x;
  return x;
}

inline ll pf(ll i) {
  return i * i;
}

const ll INF = 0xffffffff, N = 600;
bool e[N][N];
int n, sp[N][N], tot, ma, ans[N], fa[N], cnt, Ti;
ll vis[N], ec[N][N], ai, aj, mec, eii[N], a[N], q[N*N], maxq = -114514;
vector<ll>eo[N], ei[N], bp, bv;
```

```
ll ni, nj;

int ff(int p) {
  if(fa[fa[p]] != fa[p])
    fa[p] = ff(fa[p]);
  return fa[p];
}

inline void link (int i, int j) {
  fa[ff(i)] = ff(j);
  return;
}

inline void bfs(int ori) {
  bp.clear();
  bv.clear();
  memset(vis, 0, sizeof(vis));
  bp.push_back(ori);
  vis[ori] = INF + 1;
  for(int t = 0; t < bp.size(); ++t) {
    xh(i, 0, eo[bp[t]].size())
    if(vis[eo[bp[t]][i]] == 0) {
      vis[eo[bp[t]][i]] = vis[bp[t]];
      sp[ori][eo[bp[t]][i]] = sp[ori][bp[t]] + 1;
      bv.push_back(bp[t] * INF + eo[bp[t]][i]);
      bp.push_back(eo[bp[t]][i]);
    } else if(sp[ori][eo[bp[t]][i]] == sp[ori][bp[t]] + 1) {
      bv.push_back(bp[t] * INF + eo[bp[t]][i]);
      vis[eo[bp[t]][i]] += vis[bp[t]] - 1;
    }
  }


  for(; !bv.empty(); bv.pop_back()) {
    ni = bv.back() / INF, nj = bv.back() % INF;
    vis[ni] += vis[nj] % INF;
    ec[ni][nj] += (vis[nj] % INF) * (vis[ni] / INF);
  }

  return;
}

int main() {
  printf("加速倍数: ");
  rd(Ti);
  printf("人数: ");
  rd(n);
  freopen("cppData1.txt", "r", stdin);
  freopen("fha_pro10.txt", "w", stdout);
```

```
  xh(i, 0, n)
  xh(j, 0, n) {
    rd(e[i][j]);
    if(i == j)e[i][j] = 0;
    if(e[i][j]) {
      eo[i].push_back(j);
      ei[j].push_back(i);
      ++cnt;
    }
  }
  tot = cnt;
  // 读入邻接矩阵

  for(; cnt > 0; --cnt) {
    memset(eii, 0, sizeof(eii));
    memset(a, 0, sizeof(a));
    mec = -1;
    xh(i, 0, n)
    fa[i] = i;
    // 初始化

    xh(i, 0, n)
    xh(j, 0, eo[i].size())
    link(i, eo[i][j]);
    // 使用并查集连接社群

    xh(i, 0, n)
    xh(j, 0, n)
    if(e[i][j])
      if(ff(i) == ff(j)) {
        eii[fa[i]] += 2;
        a[fa[i]] += 2;
      } else {
        ++a[fa[i]];
        ++a[fa[j]];
      }
    // 计算ai与eii

    xh(i, 0, n)
    if(a[i])
      q[tot - cnt] += (ll)eii[i] * (tot << 1) - pf(a[i]);
    // 计算q的分子

    if(q[tot - cnt] > maxq) {
      maxq = q[tot - cnt];
      ma = tot - cnt;
      xh(i, 0, n)
      ans[i] = fa[i];
    }
```

```
    // 更新最优解

    if((tot - cnt) % Ti == 0) {
      memset(ec, 0, sizeof(ec));
      memset(sp, 0, sizeof(sp));
      xh(i, 0, n) bfs(i);
    }
    // 条件使用bfs计算边介数

    ai = aj = -1;
    xh(i, 0, n)
    xh(j, 0, n) {
      if(mec < ec[i][j] && ei[j].size() + eo[j].size() > 1 && ei[i].size() +
eo[i].size() > 1) {
        ai = i;
        aj = j;
        mec = ec[i][j];
      }
    }
    // 寻找边介数最大的边且这条边不会是一个点的最后一条边
    if(ai == -1 || aj == -1)
      break;
    // 若无边可删则退出循环

    ec[ai][aj] = 0;
    xh(i, 0, eo[ai].size())
    if(eo[ai][i] == aj) {
      eo[ai].erase(eo[ai].begin() + i);
      break;
    }
    // 删除此边

    xh(i, 0, ei[aj].size())
    if(ei[aj][i] == ai) {
      ei[aj].erase(ei[aj].begin() + i);
      break;
    }

    printf("t%d q%lld\n", tot - cnt, q[tot - cnt]);
    // 输出过程信息
  }

  printf("%d %lld\n\n", ma, maxq);
  xh(i, 0, n)
  printf("%d %d\n", i + 1, ans[i] + 1);
  // 输出最优解
  return 0;
}
```

附录二

计算枢纽人物和中心人物

```
#include <bits/stdc++.h>
#define xh(i,a,b) for(register int (i)=(a);(i)<(b);++(i))
#define ll long long
using namespace std;

template <typename T>
T rd(T& x) {
  x = 0;
  bool f = 1;
  char c = getchar();
  while(c > '9' || c < '0') {
    f = c - '-';
    c = getchar();
  }
  while(c >= '0' && c <= '9') {
    x = x * 10 + c - '0';
    c = getchar();
  }
  x = f ? x : -x;
  return x;
}


const ll INF = 0x3f3f3f3f, N = 550;
string nam[N];
bool e[N][N];
int n, sp[N][N], ma, ans[N], fa[N], cnt;
ll nc[N][N], ai;
vector<ll>eo[N], bp, soc[N];
ll mem[N], bs, ss;
double mbc, bc[N], cc[N], mcc;
bool by[N], sn[N];


inline void bfs(int ori) {
  bp.clear();
  bp.push_back(ori);
  nc[ori][ori] = 1;
  for(int t = 0; t < bp.size(); ++t) {
    xh(i, 0, eo[bp[t]].size())
    if(nc[ori][eo[bp[t]][i]] == 0) {
      nc[ori][eo[bp[t]][i]] = nc[ori][bp[t]];
      sp[ori][eo[bp[t]][i]] = sp[ori][bp[t]] + 1;
      bp.push_back(eo[bp[t]][i]);
    } else if(sp[ori][eo[bp[t]][i]] == sp[ori][bp[t]] + 1) {
```

```
      nc[ori][eo[bp[t]][i]] += nc[ori][bp[t]];
    }
  }
  return;
}



int main() {
  n = 500;
  memset(sp, 0x3f, sizeof(sp));
  xh(i, 0, n)sp[i][i] = 0;

  // freopen("Top500.txt", "r", stdin);
  printf("把500个人的名字贴进来：");
  xh(i, 0, n)
  cin >> nam[i];

  freopen("final.txt", "w", stdout);
  freopen("fha_pro1.txt", "r", stdin);
  xh(i, 0, n) {
    rd(fa[i]);
    rd(fa[i]);
    --fa[i];
  }

  freopen("cppData1.txt", "r", stdin);
  xh(i, 0, n)
  xh(j, 0, n) {
    rd(e[i][j]);
    if(i == j)e[i][j] = 0;
    if(e[i][j]) {
      eo[i].push_back(j);
      ++cnt;
    }
  }

  xh(i, 0, n) bfs(i);

  xh(k, 0, n)
  xh(i, 0, n)
  xh(j, 0, n)
  if(sp[i][j] == sp[i][k] + sp[k][j] && i != k && j != k && nc[i][j] != 0)
    bc[k] += (double)nc[i][k] / nc[i][j] * nc[k][j];

  xh(i, 0, n) {
    ++mem[fa[i]];
    soc[fa[i]].push_back(i);
  }
```

```
xh(i, 0, n)
if(mem[i] > 3)++bs;
else if(mem[i] == 2 || mem[i] == 3)++ss;

xh(i, 0, n)
if(mem[fa[i]] <= 3)by[i] = 1;

xh(i, 0, bs) {
  mbc = -114514;
  xh(j, 0, n)
  if(bc[j] > mbc) {
    mbc = bc[j];
    ai = j;
  }
  bc[ai] *= -1;
  sn[ai] = 1;
}

printf("枢纽：\n");
xh(i, 0, n) {
  if(sn[i])
    cout << i + 1 << ' ' << nam[i] << " " << bc[i] * -1 << '\n';
}

printf("边缘人物：\n");
xh(i, 0, n) {
  if(sn[i] == 0 && by[i] == 1)
    cout << i + 1 << "  " << nam[i] << '\n';
}

printf("中心人物：\n");
xh(i, 0, n)
if(soc[i].size() > 3) {
  mcc = -1;
  printf("社群%d：", i + 1);
  xh(j, 0, soc[i].size()) {
    xh(k, 0, soc[i].size())
    if(sp[soc[i][j]][soc[i][k]] != INF)
      cc[soc[i][j]] += sp[soc[i][j]][soc[i][k]];
    cc[soc[i][j]] = (soc[i].size() - 1) / cc[soc[i][j]];
    if(cc[soc[i][j]] > mcc) {
      mcc = cc[soc[i][j]];
      ai = soc[i][j];
    }
  }
  cout << ai + 1 << "  " << nam[ai] << " " << cc[ai] << '\n';
}
```

```
  printf("全局中心人物: \n");
  memset(cc, 0, sizeof(cc));
  ai = mcc = -1;
  xh(i, 0, n) {
    xh(j, 0, n)
    if(sp[i][j] != INF)
      cc[i] += sp[i][j];
    cc[i] = (n - 1) / cc[i];
    if(cc[i] >= mcc) {
      mcc = cc[i];
      ai = i;
    }
  }
  cout << ai + 1 << "  " << nam[ai] << " " << cc[ai] << '\n';
  return 0;
}
```

附录三

计算结果

```
枢纽:
5 神楽七奈Official 1054.66
10 小狼XF 858.078
33 菜菜子Nanako 3243.67
48 帕里_Paryi 2597.68
52 犬山玉姬Official 2689.42
60 疾風醬 1418.78
67 琉绮Ruki 2060.55
91 QQ星 1183.72
96 早见咲Saki 1420.25
103 時雨羽衣Official 984.888
113 安晴Ankii 1915.88
117 不举萌栗 913.888
139 筱魈Syou 835.55
149 龙胆尊Official 860.129
174 まこと-macoto官方 1073.39
179 赫卡Tia 2780.14
195 田汐汐_Official 1427.09
204 花留Karu 1164.92
232 魔宫永恋Mamiya 827.165
292 露珀塔_Ruperta 2085.43
```

```
边缘人物:
11  一块电鹿板
17  AIChannel中国绊爱
30  有栖Mana_Official
```

45 猫宫日向Official
50 是姬拉Kira
72 葛叶Official
81 本间向日葵Official
84 兮子cc
94 叫我默默酱
100 ?Sakulaˇ小舞
107 九重紫Official
109 心斋桥オクトOfficial
110 超想吃番茄
123 猫芒ベル_Official
130 阳向心美Official
131 凛-阿格里亚斯
134 扇宝
137 雨街F
147 鹿野灸
153 椎名菜羽Official
157 眞白花音_Official
160 未来明-MiraiAkari
162 修女克蕾雅Official
188 希月萌奈
189 三日曆Official
191 里奈Rina
193 勾檀Mayumi
205 郡道美玲Official
210 安堂いなり_official
211 温之九Dolores
223 弥希Miki
224 dodo_Official
228 召唤师yami
233 神奈希_Official
234 星汐Seki
248 火羽ひのめ
250 画外灯official
253 中单光一
254 小铃久绘Official
256 罗菈Rola
257 奈奈莉娅Channel
259 丝茉茉i
264 艾尔莎_Channel
265 南宫灯official
266 宇佐紀ノノ_usagi
268 龙西Rinya
272 永远永远酱w
273 无前Namae
274 绫濑光Official
277 北柚香Yuka
280 诺莺Nox
287 果汁erR

299 五月织姬ctsu
301 福山Master
306 莉姆丝OvO
307 AIofficial
310 黑桐亚里亚Official
311 坏宝Huaib
314 眠眠official
318 羽咲Rabi_Official
320 龙包包
325 相羽初叶_Official
328 Kirikosama_
329 烤鱼子Official
337 火花工作室
338 沙月ちゃん
345 惑姬Waku
361 伊莎贝拉Official
362 沙夜_Saya
367 晴奈子_Senako
369 尼奥喵Neil_Channel
374 珞璃Official
376 漪月灵-Aether-
380 咕咕挞Official
382 佐娜Sanya
386 月兔华-WACTOR官方
388 许亦真_REAL
391 真绯瑠mahiru
402 木暮Piyoko-官方
405 辛西娅_Cynthiaa
418 呆呆喵
420 紫海由爱Channel
423 纸代棗-やすゆきOfficial
434 AI艾灵
441 库莉姆Cream
442 kinoko杂菌
444 亚可Ako
448 咿籽芥末大尾巴狼itsu
449 恋诗夜Koxia
452 云玉鸾
454 OscarAnd奥斯卡
460 栗子酱RIKO
461 郡道美玲的养豚厂
463 Okome_VTuber官方频道
474 露露娜Ruruna
477 天川花乃official
481 立花遥はるか
482 洛佩sama
485 蛙吹Keroro
488 夏卜卜
491 DJGun

```
493  万灼
494  佳茵是个小富婆
```

```
中心人物:
社群98: 2  hanser inf
社群125: 9  C酱です inf
社群138: 138  鈴宮鈴 inf
社群170: 170  紗耶_sayako inf
社群187: 187  凤玲天天Official inf
社群212: 212  花满Official inf
社群221: 128  古守血遊official inf
社群246: 13  HiiroVTuber 15.8
社群258: 258  艾露露Ailurus inf
社群336: 336  季毅Jiyi inf
社群339: 124  ywwuyi inf
社群349: 349  竹花诺特Official inf
社群401: 401  星野琳奈 inf
社群412: 412  病院坂Rei inf
社群422: 422  魔法少女真理酱 inf
社群430: 430  卡欧斯_Chaos inf
社群446: 446  喵月nyatsuki inf
社群453: 371  尤狸Yuriri inf
社群492: 203  喵奈Official inf
社群499: 8  嘉然今天吃什么 inf
```

```
全局中心人物:
2  hanser 71.2857
```

附录四

虚拟主播星系图数据连边脚本

```
const fs = require('fs')
const { resolve } = require('path')

const groupData = require('../../results/charts/group.json')
const hingeData = require('../../results/charts/hinge.json')
const centerData = require('../../results/charts/center.json')
const aloneData = require('../../results/charts/alone.json')

const indexMap = require('../../results/attactions/cpp-data/lessData/indexMap.json')
const indexArray = require('../../results/attactions/cpp-
data/lessData/indexArray.json')
const connData = require('../../results/attactions/cpp-data/lessData/cppData.json')

const chartData = { nodes: [], links: [] }

const vtbNameList = indexArray
```

```
// (1) nodes
const renderName = vtbName => {
  return centerData.includes(indexMap[vtbName])
    ? vtbName + '_中心人物'
    : aloneData.includes(indexMap[vtbName])
      ? vtbName + '_边缘人物'
      : vtbName
}
vtbNameList.forEach(vtbName => {
  chartData.nodes.push({
    id: '' + indexMap[vtbName],
    name: renderName(vtbName),
    val: centerData.includes(indexMap[vtbName])
      ? 380 : aloneData.includes(indexMap[vtbName]) ? 10 : 60,
    group: centerData.includes(indexMap[vtbName])
      ? 3 : aloneData.includes(indexMap[vtbName])
        ? 1 : hingeData.includes(indexMap[vtbName]) ? 4 : 2
  })
})

// (2) links
vtbNameList.forEach(vtbName => {
  const vtbIndex = indexMap[vtbName]
  connData[vtbIndex].forEach((val, index) => {
    if (val === 1) {
      // vtbIndex -> index
      if (groupData[vtbIndex] === groupData[index]) {
        chartData.links.push({
          source: '' + vtbIndex,
          target: '' + index,
          width: 1,
          length: 200
        })
      } else {
        chartData.links.push({
          source: '' + vtbIndex,
          target: '' + index,
          width: 1,
          length: 3300
        })
      }
    }
  })
})

fs.writeFileSync(
  resolve(__dirname, '../../results/charts/chartData.json'),
  JSON.stringify(chartData, null, 2)
```

```
)
fs.writeFileSync(
  resolve(__dirname, '../../../analyse/charts/ex/data.js'),
  'var data = ' + JSON.stringify(chartData)
)
```

附录五

关系图图脚本

```
const options = {
  controlType:'trackball',
  extraRenderers: [new THREE.CSS2DRenderer()]
}
const ifm = document.getElementById('graph')
ForceGraph3D(options)(ifm)
  .graphData(data)
  .nodeLabel('name')
  .nodeAutoColorBy('group')
  .linkDirectionalParticles(link => link.width)
  .linkDirectionalParticleSpeed(.01)
  .linkWidth(link => link.width / 5)
  .linkOpacity(.4)
  .linkAutoColorBy(link => link.source)
  .linkDirectionalParticleWidth(link => link.width / 5)
  .d3Force('link')
  .distance(link => link.length)
```