

# 图的可视化分析

## ——对哔哩哔哩弹幕视频网 2019 年“百大 UP 主”的关系分析

### 1. 选题说明

十多年前还是小众的亚文化社区的[哔哩哔哩弹幕视频网](https://www.bilibili.com/) (以下简称 B 站), 自从 2018 在纳斯达克上市后发展越来越壮大, 用户基数和日活跃用户数等数据不断上涨, 许多例如央视新闻这样的官方平台也开始入驻。B 站优质内容的主要生产者是上传原创视频的“UP 主”, 从 2018 年开始 B 站都会评选出当年最具有影响力的 100 位 UP 为当年的“百大 UP 主” (以下简称百大)。作为 B 站的资深用户, 本人也关注了百大中的许多优质 UP 主, 对于探究各个优秀 UP 主的关系十分感兴趣, 所以选择了这个题目。另外探究百大的关系或许可以增进对 B 站社区文化的了解, 挖掘出平常注意不到的信息。

### 2. 数据收集

#### 2.1 收集思路

由于没有现成的数据, 需要用爬虫来抓取一手数据。<sup>1</sup>

爬虫使用 python 编写 (见文件 main.py), 利用 b 站提供的 api 获取信息。Api 是从 github 项目“bilibili-API-collect”中获取的 (项目地址: <https://github.com/SocialSisterYi/bilibili-API-collect>)。

#### 2.2 收集方法

---

*prepare: api, Stack or Queue, 百大 uid*

*Stack/Queue  $\leftarrow$  百大 uid.*

*While Stack/Queue not empty:*

*up  $\leftarrow$  Stack.top/Queue.front.*

*get follow\_list of up*

*储存下 up 即其关注者信息.*

*for follow in follow\_list:*

*If follow not visited:*

*Stack/Queue  $\leftarrow$  follow.*

---

获取百大 uid 的方式是首先注册一个 b 站账号, 关注所有 2019 年百大 up 主, 通过 api 读取这个账号关注的所有人, 就可以得到百大的 uid 了。这个爬取过程本身是一个图构建的过程: 每个 uid 对应一个 up 主, 可以作为结点, 利用邻接表的方式储存边 (即关注) 的信息, 最终会得到一个有向图, 也就是说图的结构在爬虫阶段就能自动构成。

#### 2.3 收集的局限性

上述过程在实践中出现了问题, 在实践中, 获取了 10000 个 up 主的信息后, 队列/栈

---

<sup>1</sup> 注意, 爬取数据的时间是五月初, 到现在可能关注信息有变, 但影响不会太大

中还有 70000 以上的 uid，且队列和栈的大小还在不断增加，没有收敛的迹象，说明我最后至少可以得到 80000 个 up 主的信息，但这些人中有许多并不是知名的 up 主（粉丝数少，影响力低），若在爬虫的过程中就根据粉丝数进行筛选会使效率较低，由于频繁访问我的请求也被服务器拒绝了。

在实验的时候我使用了广搜和深搜两种方法，但鉴于让这个获取过程收敛是很难的，所以决定不使用深度搜索的思路，只使用广度优先搜索，优先搜索与百大 up 主关系较近的，当搜索到一定人数时停止。

## 2.4 数据获取的结果

最终得到了 15000 位 UP 主的名称，分区于关注者信息用于分析。在文件 up\_meta.json 中。

## 3 数据预处理与图构建

### 3.1 数据格式处理

进行图的算法编写时我更倾向于使用 C++，所以先将得到的原始数据转化为方便读取的格式。这一步还要：顺便筛选掉一部分没有关注任何人的 UP，因为没有关注任何人的话在图构建中是用不上的；把每个 UP 对应的分区转化成数字编号，方便分析。

把符合条件的 UP 主信息存入一个文本文件中，格式为：第一行是总人数，从第二行开始，每行为该行 up 的 uid+空格+up 名称+空格+以空格分隔的关注者 uid+0。当读到 0 时就结束一位 UP 主信息的读取。（数据处理代码在 convert\_data.ipynb 中，得到的数据为 up\_info\_utf8.txt，由于 C++ 处理 utf-8 比较困难，所以把编码调整后得到 up\_info.txt）

最终用于图构建的数据如下：

```
14444
562374290 test 0 7349 63231 116683 168598 185546 282994 284120 322892 375375 423895 515993 562197 777536 927587 946974 1565155 1577804
7349 STN工作室 1 5970160 51705359 43536 37663924 489535836 419588355 8990248 326081112 946974 85514191 176037767 397540531 32708316 4311
63231 泛式 2 24801003 90596242 19334905 381016 2196632 353840826 517327498 112938490 312001 9034870 83953807 113362335 27880221 1328260
116683 =咬人猫= 3 5197192 654839 99673 119418 109 259333 8084905 4350178 433715 622863 271126 179628 0
168598 逍遥散人 1 10922568 16391597 433351 419220 28259096 8679195 2899425 8148923 4408538 46804148 12270670 254463269 44473221 1774758
185546 小可儿 4 523981079 430510309 358606 17474339 103251824 297189227 10330740 883968 15371308 5848380 35719553 2853209 18690024 16506
282994 冷鸢yousa 4 1871297 10007772 417583826 430801209 454719565 37663924 213059082 315898473 364225566 80333183 486633990 80900494 50
284120 三无Marblue 4 345630501 90942983 16554770 396194 417583826 53456 368958 1467772 472747194 262453663 483203976 3046429 430969192
322892 痒局长 1 37946996 174501086 7552204 326499679 808171 423895 234256 466272 419220 183430 67141 4162287 43536 168598 7487399 391675
375375 伊丽莎白鼠 5 11332884 53456 472747194 163637592 43855 647950 235555226 227933 8047632 15067942 320772967 7792521 777536 6355324 1
423895 怕上火暴王老菊 1 100459364 80509102 485092296 216156027 13839178 1932102 9923991 120122306 383578614 551770 409096076 382193067 3
515993 枪弹轨迹 5 30222764 250648682 13248198 233658087 357229416 6171645 383444910 38640369 129681040 394071338 96070394 22141287 80341
562197 中国BOY超级大猩猩 1 35359510 3066511 2403047 5970160 25876945 43976732 328135510 12807175 63231 416128940 19526512 777536 2206456
777536 LexBurner 2 381016 2206456 13354765 546195 562197 59905809 466272 9824766 5626845 1577804 208259 7558565 168598 26311441 4079592
927587 木鱼水心 6 33382000 2374194 1935882 883724 3379951 284120 374692347 423442 20165629 25876945 294081438 387982605 32708692 1577336
946974 影视飓风 7 43646124 17546432 570820 116683 430510309 33382000 50212909 254463269 517327498 390461123 86273007 218484579 437316738
1565155 起小点是火腿 1 1558103 429896899 113362335 437316738 503569552 7552204 7792521 390461123 37663924 305214791 258150656 495695169
1577804 某幻君 1 163637592 2403047 17409016 50329118 185546 12807175 22227 777536 75 161419374 389088 177230427 6585913 659965 24295808
1643718 山下智博 8 393166851 295723 7552204 919244 437316738 268990278 6574487 441970571 299263368 434705993 2306780 321173469 339104478
1950209 水一大魔王 1 12333557 347728422 17819768 195223577 123938419 122879 0
2374194 墨韵Moyun 1 257975429 385200931 12373184 29153009 61287828 9905808 260452858 517327498 278761627 25422790 27756469 40607244 5260
3380239 神奇的老皮 8 3211302 435387918 526559715 48033476 33151008 54992199 488423046 2609425 5816073 472747194 4162287 27534330 4523093
3766866 科技美学 7 17474339 396895652 23020989 517327498 394205865 957060 7792521 13839125 356652866 328321630 69770725 11257503 123372
3957971 贤宝宝Baby 8 37663924 7026560 3125954 416128940 6574487 1393437 37090048 5970160 12807175 21463726 99015667 4263163 2920960 7456
4162287 梁博之C君 1 433351 176037767 684169 7391373 562197 43536 256763 15834498 351007 2976992 901072 3337724 0
```

### 3.2 图构建

在 2.2 中说到在爬虫过程中就可以得到一个图结构，但是这个图结构肯定不是我们想要的，因为从现实来看，仅有每个 up 主关注的人并不足以判断他们是否认识还是单方面关注。但如果知道两个人相互关注，那我们基本可以判定他们是认识的。所以最终图构建的方法是，遍历每个 up 关注的人，若他关注的人也关注了他，就认为他们之间有边。如果一个 UP 和任何人之间都没有边，就把他去除。最终图中还留有 4560 个节点。

仅知道两个节点间有边是不够的，还需要知道权重。比如我要知道怎样最快的从一个 UP 主开始联系上一些 UP 主来参与活动，就需要得到最小支撑树。那么我应当设定权重越小时



两人关系更密切。最近 B 站有推出联合投稿功能，最好的方式是用联合投稿的次数来衡量关系密切程度，但是我并没有爬取联合投稿次数的方法，因此只能退而求其次，认为两个人共同关注的 UP 数量多就说明他们关系密切。这样的假设也是有合理性的，比如有几个 UP 主都是好朋友，他们之间都互相关注（他们几个在一个完全子图中），那么他们共同关注的数量肯定是大的。所以我们把共同关注的人数取倒数当作边权（边权范围(0,1]），若两个人没有共同关注就把边权置为 2。

## 4 图算法

### 4.1 最短路径：

对于 UP 主的最短路径，其现实意义是当一个 UP 主要与另一个合作时最短通过几个人可以找到他。在这个算法中我不打算考虑他们之间的权重，实现了一个不带权重的 Dijkstra 算法。算法的实现在函数 `Node* find_way(int up1,int up2)` 中，使用时调用 `void query_way()`，在控制台输入想查询的两个 UP 主的 uid，会输出他们之间的路径，一次查询结果如下：

查找 UP 主中国 BOY 和半佛仙人间的路径：

```
输入两个up的uid，查找路径:562197 37663924
中国BOY超级大猩猩 特效小哥studio 陈睿 硬核的半佛仙人
```

### 4.2 连通分量：

这个算法的实现是比较简单的，实现方法就是用广搜或者深搜中的任意一种方法搜索这个图到收敛，然后再遍历所有结点看有没有未被访问的，再从这个未被访问的开始搜索…每搜索一轮，都把节点储存在一个子图中(结构体 Graph)，这样我们可以通过读取每个 Graph 来展示一个连通分支的结构，或者求这个连通分支的最小支撑树。求连通分量的算法实现在函数 `get_branch()` 中，所有分支都会被存在 `vector<Graph*>branch` 中。

从我的数据中求出了 42 个连通分支，最大的连通分支有 4466 个节点，占总数的 97.9%。其他的连通分支有 2-5 个节点不等。

### 4.3 最小生成树

利用 4.2 中计算得到的各个分量和 3.2 中的权重，我们可以计算出每个连通分量的最小生成树，对一个 Graph 调用了 `prim(Graph* G)` 后，可以用 Graph 的方法 `show_tree()` 在控制台来展示最小生成树的结构，最大的生成树结构如下：

```
本末测评<-STN工作室 黑桐谷歌<-STN工作室 毒德大学字幕组<-STN工作室 豆豆子ex<-STN工作室 Gamker攻壳<-STN工作室 昨天我们攻
了大书库<-STN工作室 PS4游戏姬<-STN工作室 盛嘉成<-STN工作室 星际老男孩<-歪杰 李老鼠说车<-STN工作室 2KGames中国<-STN工作室
CD_PROJEKT_RED<-STN工作室 芒果冰OL<-STN工作室 育碧中国Ubisoft<-STN工作室 酷游掌中报<-STN工作室 龙崎棒棒糖<-STN工作室 飞
燕群岛<-STN工作室 哆哩糊途<-STN工作室 愁我是智障<-STN工作室 Fuji 玫瑰叔<-STN工作室 大笨象Vito<-STN工作室 游民星空官方<-S
TN工作室 暴雪游戏动力<-STN工作室 梦似清酒人未休<-STN工作室 TBS官方频道<-STN工作室 Animenzzz<-STN工作室 努力的Lorre<-ST
N工作室 凹凸君说<-STN工作室 靠脸吃饭的徐大王<-STN工作室 菊子桑<-STN工作室 我是怪异君<-STN工作室 教父桑<-STN工作室 水蛭-J
ogsLeech<-STN工作室 极客湾Geekwan<-STN工作室 吃素的狮子<-STN工作室 逗川kshadow<-STN工作室 黑椒墨鱼<-STN工作室 mo6638<-
STN工作室 水蛭 JogsLeech<-STN工作室 小绝<-STN工作室 起小点是大腿<-STN工作室 哦漏QAQ<-STN工作室 天天卡牌<-STN工作室 半半
睡<-STN工作室 别让游戏玩人<-STN工作室 朵、天堂<-STN工作室 抽风的飞机工作室<-STN工作室 鸡鸡·夫斯基<-STN工作室 LexBurner<
-STN工作室 尚在天国EX<-STN工作室 Stir<-STN工作室 黑猫厨房<-本末测评 老将巨靠谱<-本末测评 兔爹哭晕在厕所<-本末测评 伟嘉ga
ra616<-本末测评 answer824<-本末测评 大佬甜er<-本末测评 以安噜噜喂<-本末测评 福乐小哥<-本末测评 Rainie田<-本末测评 摄影
师本舟<-本末测评 本末同学mumu<-本末测评 本末测评许亚军<-本末测评 哪怪不喝油<-黑桐谷歌 能人族Official<-黑桐谷歌 随义のfre
ely<-黑桐谷歌 Diriya伯爵<-黑桐谷歌 李令羽<-黑桐谷歌 EdmundDZhang<-黑桐谷歌 矮乐多Aliga<-黑桐谷歌 女王盐<-黑桐谷歌 篝火营
地视频<-黑桐谷歌 叉空<-黑桐谷歌 欣小萌<-黑桐谷歌 予言姐姐<-黑桐谷歌 enya甜<-黑桐谷歌 瓶子君152<-黑桐谷歌 千槊型DJ<-黑
桐谷歌 赖耳<-黑桐谷歌 游戏机实用技术UCG<-黑桐谷歌 一根香肠君<-黑桐谷歌 Xpeak | 小峰、<-黑桐谷歌 林大B<-黑桐谷歌 麦子麦
好<-黑桐谷歌 极游组<-黑桐谷歌 梦之城阿狸<-黑桐谷歌 神奇陆夫人<-黑桐谷歌 DMYoung<-黑桐谷歌 阿良良木健<-黑桐谷歌 小野道ono
<-黑桐谷歌 皮卡洁Yanako<-黑桐谷歌 PlayStation中国<-黑桐谷歌 yakkol陪影<-黑桐谷歌 太懒要觉觉<-黑桐谷歌 CarlyLee<-黑桐谷歌
慕染音弦<-黑桐谷歌 蛋黄姬GAT-X105<-黑桐谷歌 杨小波<-黑桐谷歌 懒猪欣<-黑桐谷歌 庄不纯<-黑桐谷歌 jackdaw0001<-黑桐谷歌 Lum
yu<-黑桐谷歌 猫耳白丽灵梦<-黑桐谷歌 人同失<-黑桐谷歌 游戏篝火营地<-黑桐谷歌 小井Roi<-黑桐谷歌 新月冰冰<-黑桐谷歌 野食
小哥<-黑桐谷歌 重果仁研究协会<-黑桐谷歌 陈睿<-黑桐谷歌 夏日的花芽<-黑桐谷歌 视角姬<-黑桐谷歌 小由伊<-老骚豆腐 小宇点z<-
黑桐谷歌 鱼炒饭<-黑桐谷歌 雪夜物语<-黑桐谷歌 Q-kun<-黑桐谷歌 大谷的游戏创作小屋<-黑桐谷歌 痒局长<-黑桐谷歌 doraiba<-黑桐
谷歌 我的样子平平无奇<-黑桐谷歌 晓南<-黑桐谷歌 掌机王SP苍穹<-黑桐谷歌 赤九玖<-黑桐谷歌 罗兹<-黑桐谷歌 渗透之C君<-黑桐谷
歌 Alessa0<-黑桐谷歌 管家厨波里斯<-黑桐谷歌 一个迟迟<-黑桐谷歌 鹹湿老骗纸<-黑桐谷歌 lol今天玩什么<-黑桐谷歌 空耳YAYA<-黑
桐谷歌 皮蛋<-黑桐谷歌 机核网<-黑桐谷歌 怕上火暴王老菊<-黑桐谷歌 麦教授<-黑桐谷歌 团鼠君<-黑桐谷歌 乌鸦karasu<-黑桐谷歌
天刀罪歌<-黑桐谷歌 hyun310<-黑桐谷歌 SolidEx<-黑桐谷歌 女流66<-黑桐谷歌 木须柄Alexpad<-黑桐谷歌 祈Inory<-黑桐谷歌 飞剑<
黑桐谷歌 Miraino 阿凌<-黑桐谷歌 碧诗<-黑桐谷歌 司波图<-毒德大学字幕组 摄影师顾俊<-毒德大学字幕组 阿豪的阁楼<-毒德大学字
幕组 极速拍档<-毒德大学字幕组 Alex 梁<-毒德大学字幕组 Jacob 裤子<-毒德大学字幕组 拍胶片的新疆仔<-毒德大学字幕组 彩峰摄影
```

这个显示方式比较难看，之后可视化部分会有可视化的显示。

#### 4.4 中心度

实现了计算了紧密中心度，方法比较粗暴，直接计算一个节点到其它节点的最短路径之和，就可以得到紧密中心度。在函数 `cal_closeness()` 中。

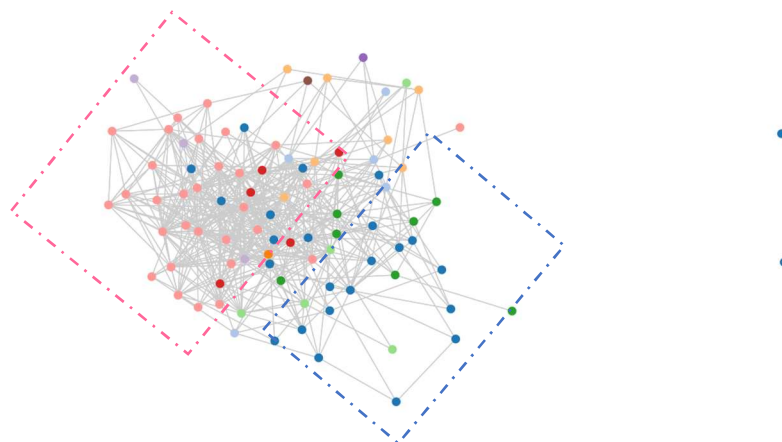
### 5 可视分析

#### 5.1 百大 UP 关系图

选题初衷就是探究百大 UP 的关系，之所以除了百大 UP 之外搜集一些别的相关的 UP 主是为了获取更多的信息用于分析，比如给出权重或者进行路径查询等等。把 4560 个节点全部都进行可视化是很难的，所以我先仅可视化百大 UP 的关系图，使用 `d3.js` 可视化结果在 `graph_big_100.html` 中：

#### 百大up关系图

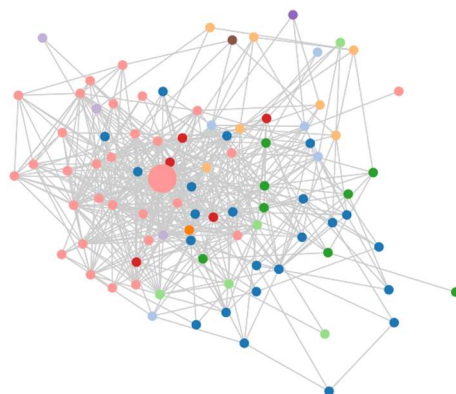
选择节点查看详细信息



在鼠标移到某一个节点上时节点会放大，并且在上方的文字会显示该 UP 的一些信息，可以点击链接进入该 UP 主的 B 站空间查看详情，如下图：

#### 百大up关系图

Up主名称: [歪果仁研究协会](#), 分区: 生活



为了了解各个分区之间 UP 主的关系, 根据不同的分区给 UP 主对应的节点染上了颜色, 如果几个节点之间互相连接的边较多, 那么这些节点很容易聚集到一起, 他们间的连边也会显得很密集。如图所示, 粉色, 蓝色这两种数量最多的颜色有明显的聚集, 粉色为生活区, 蓝色为游戏区, 说明这两个分区的 UP 主内部关系是比较紧密的, 实际上他们确实会经常合作发布视频或者参加活动。

也可以发现其实粉色和蓝色两种颜色的节点之间的联系也很密切, 有些蓝色节点几乎打入了粉色节点的“内部”, 有可能是这两个分区之间互动频繁。可以查看几乎进入粉色内部的蓝色节点代表的 UP 主有“中国 BOY”和“小潮院长”等, 查看他们的投稿, 中国 BOY 和小潮院长的生活区投稿都占到了总投稿的 20%以上, 他们也与主攻生活区的 UP 主有着合作。

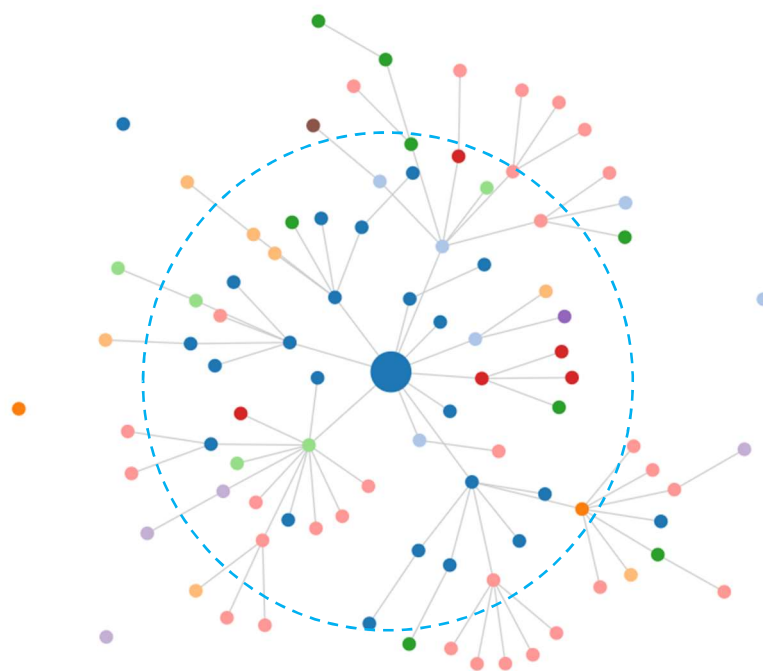
通过以上分析, 可视化的图结构确实可以帮助发掘 UP 主的信息, 经过验证这些分析都是正确的。如果将更多的节点加入图中进行可视化, 可能可以发掘出更多的信息。

## 5.2 最小生成树

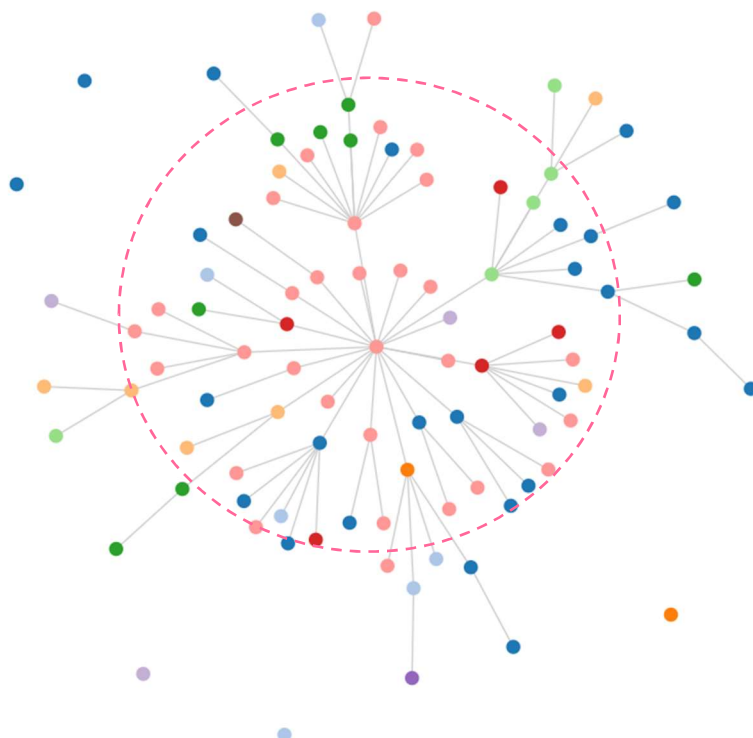
同样我可视化了百大间的最小生成树, 我进行了两次 prim 算法, 一次是以一位游戏区

# 百大up最小生成树

Up主名称:STN工作室,分区:游戏



UP 主开始进行, 另一次是以一位生活区 UP 主开始进行, 会得到两棵不同的最小生成树如下



最小生成树可视化界面的交互方式与 5.1 种图的界面相同。

本身最小生成树就不一定是唯一的，因为最小生成树算法只保证了边权之和是最小的，由于我的边权计算方式是共同关注数，所以出现很多边边权相同也是正常的。

可以看到若以游戏区（蓝色）UP 主为起点开始生成，蓝色节点都会占据中心的位置，以粉色节点为起点那么粉色节点就会占据中心位置，且可以发现蓝色节点“围绕”着粉色节点，也可以看出游戏区与生活区两个分区之间的关系密切。

### 5.3 紧密中心度





可视化之后发现紧密中心度的意义并不怎么大，在计算最短路径的时候我也发现了取两个百大 UP 主之间的路径，长度不会有特别大的区别，一般来说路径长为 5 左右就是比较长的了，说明百大 UP 主之间的联系是比较紧密的，紧密中心度的差别并没有特别大，可视化出来这些圆圈的大小也区别不大。

## 6 该项目的贡献与待改进的地方

编写了可以获取 2019 百大 UP 主即其相关 UP 主信息的爬虫，编写了处理爬取得到的原始数据的脚本。对处理后的数据进行整理和图的构建，最短路径算法可以找出两个 UP 主之间最短的“联络路线”，连通分支算法把整个非连通的图分成数个连通的子图，对于每个子图都可以调用最小生成树的算法来求的最小生成树。将百大 UP 主的节点单独导出进行了可视化，可以从可视化的图中发掘信息，如发现某些分区间的合作关系等。

当然这个项目有很多不完善的地方，由于能力有限没有办法把所有节点都可视化，导致有大量的数据没有用于分析，除了百大的节点还有约 4500 个节点可以发掘更多信息；没有进一步实现提高算法等等。如果再给我一次机会我一定会找人组队，考虑到疫情线上交流不便就打算一个人做，但一个人的力量终究是有限的，有很多想法比如 Community detection 并没有成功实现（写出了无法解决的 bug）。

## 7.对于环境与文件的说明

使用的 python 版本为 3.7.4，C++ 的集成开发环境为 VS2019。

bin 文件夹内为网页的文件，index.html 包含了三个可视化的图/

src 文件夹内 graphAnalysis.sln 是 C++ 的项目文件，graphAnalysis 文件夹内有所有用到的数据，生成的数据，和生成数据的脚本，spider 文件夹内是爬虫的脚本与爬虫得到的原始数据。

## 8.参考资料

爬虫：<https://github.com/SocialSisterYi/bilibili-API-collect>

D3: <https://wiki.jikexueyuan.com/project/d3wiki/>

<https://observablehq.com/@d3/gallery>