**本代码实现基于论文：Adaptive Influence Blocking: Minimizing the Negative Spread by Observation-Based Policies（**[**https://ieeexplore.ieee.org/document/8731584**](https://ieeexplore.ieee.org/document/8731584)**）**

**编译方法**

在Linux系统上用g++编译即可，建议使用g++>=7.5.0。命令如下：

g++ -O2 \*.cpp sfmt/SFMT.c -o aiblt

**社交网络图数据集的输入和处理**

数据集以文本文件形式给出，其文件名没有特殊要求。模板如下：

**m**

**u\_1 v\_1**

**......** （此处省略m-2行）

**u\_m v\_m**

输入的社交网络图是一张有向图，根据上面格式，在数据集文件的第一行应当为一个整数，表示图中的总边数m。之后文件应当包涵m行，其中每行有两个整数u和v，表示数据集中存在有向边(u,v)。

若社交网络图有n个节点，则程序将会自动把节点标号映射为从0开始到n-1的标号，其具体做法如下：

1. 每次从数据集中读取一个节点标号x，初始时设置变量V=0。注意这里的x可能是边的起点u或者终点v，两种情况同样对待。

2. 如果x的标号未被保存，则其标号为V并保存之，然后将V加一。否则，读取x已被保存的标号。

3. 返回步骤1，直到数据集读取完。

在获得社交网络图的结构后，软件将随机生成每条边的边权，目前暂不支持用户自定义边权。为了满足线性阈值模型的边权要求，软件按照如下方法随机生成边权：

1.从配置中获取参数wsum。

2.考虑每个节点v，对于它的每一条入边生成一个随机数

3.对上述随机数进行归一化使得这些随机数之和为wsum，归一化后的随机数即为每条边的边权。

4. 返回步骤2，直到考虑完了所有节点。

可以通过设置来输出这些边权，将在后面提到。

**参数配置方法和运行方法**

参数配置通过一个文本文件进行，其文件名应该以.in结束。该文本文件的路径将作为参数传入命令行。配置文件的格式为如下所示，其中粗体表示用户需要输入的参数，而非加粗的部分表示固定的文字，仅用于表示后面的参数名称，在任何配置文件中非粗体部分都是不变的。

**path**

graphseed **graphseed**

seed **seed**

l **l**

epsilon **epsilon**

type **type**

k **k**

t **t**

wsum **wsum**

alpha **alpha**

evalcnt **evalcnt**

verbose **verbose**

ninfseeds **ninfseeds**

**infseedtype**

**s1 s2 ......** （**ninfseedtype**=0时此行省略）

下面逐一解释配置文件中每个字段（加粗部分）的含义。

1. 文件第一行应为一个字符串path，它表示数据集文件（上一节中已提到数据集的格式）的存放路径，可以使用相对或者绝对路径。

2. 第二行的graphseed必须为正整数，表示用于生成边权的随机数种子。边权生成方法已经在上一节提到。

3. 第三行的l表示IMM算法中的参数l，控制推理的精度，必须为正整数，学术界一般认为该值可以设置为1。

4. 第四行的epsilon表示IMM算法中的参数epsilon，控制推理的精度，可设置为0-1之间的实数值，一般设置为0.5一下，越小则精度越高，但同时运行时间也会显著增加。

5. 第五行的type必须为0到4之间的整数，决定运行的阻挡影响力推理算法的种类。其中0表示只进行仿真模拟，不进行推理。而1到4分别表示k-r算法、a-T算法、OS-RP算法和OS-IP算法。

6. 第六行的k必须为整数，表示阻挡影响力最大化问题中的预算参数k，前文已说明。

7. 第七行的t必须为整数，表示阻挡影响力最大化问题中的时间阈值T，前文已说明。

8. 第八行的wsum表示生成边权时的归一化参数wsum，必须为大于0小于等于1的实数。前文已说明。

9. 第九行的alpha表示a-T算法中的参数a，必须为0-1之间的实数，具体前文已说明。

10. 第十行的evalcnt表示进行社交网络影响力传播的仿真模拟时的迭代模拟次数，必须为正整数，数值越大则模拟精确度越高，同时运行时间会增加。

11. 第十一行的verbose表示是否开启额外输出，可以用于调试等目的，其值应为整数0或者1。当其值为0时，表示不输出额外输出，当其值为1时，表示将会输出额外输出。具体将在下面运行结果说明说详细描述。

12. 第十二行中的ninfseeds表示影响力传播种子的个数，应为正整数。

13. 第十三行中的infseedtype表示影响力传播种子的生成方法，其值应为整数0或者1。当其值为0时，表示软件随机生成所有种子。当其值为1时，表示将在下一行输入所有种子。

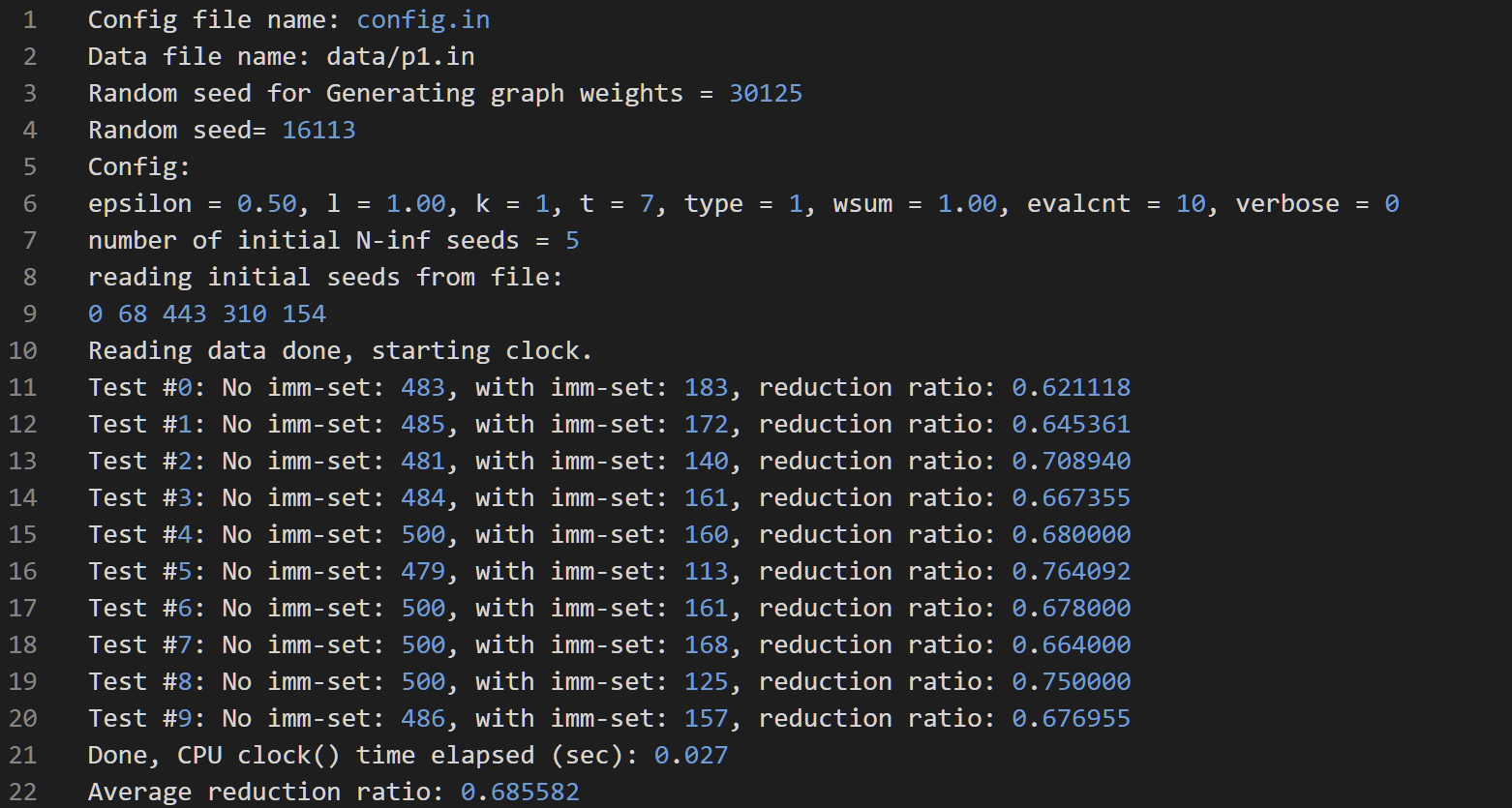
14. 第十四行中为当上一行的infseedtype=1时，输入的种子节点编号。这里的编号应为ninfseeds个以空格分割的非负整数，表示根据上文所述节点重标号方法进行重标号之后的种子节点标号。如果上一行为0，则此行输入无效，软件仍将随机生成种子。

准备好配置文件后，要运行程序，只需要在命令行中执行可执行文件（默认名称为aiblt），并且传入不含.in后缀的配置文件名作为参数，例如，假设配置文件名为config.in，且当前命令行目录就是可执行文件所在目录，则应在命令行中输入以下内容：

./aiblt config

**运行结果说明**

当上述参数verbose=0时，程序将在当前命令行目录下输出一个X.log文件作为输出，其中X表示配置文件不含.in后缀的文件名，即上述输入命令行中作为参数的字符串。例如运行上一节作为例子的的命令行后，软件将输出的文件为名config.log。



如上图所示为输出文件的一个样例，下面解释其内容：

1. 第一行中，输出了配置文件名。

2. 第二行为数据集文件名。

3. 第三到第九行输出了配置文件中的各类参数，这些参数的含义已经在前面配置文件的介绍中提及。

4. 从第十一行开始为仿真模拟结果，在该截图样例中，由于evalcnt=10，所以可以看到共有10次模拟结果。这些结果的格式如下：

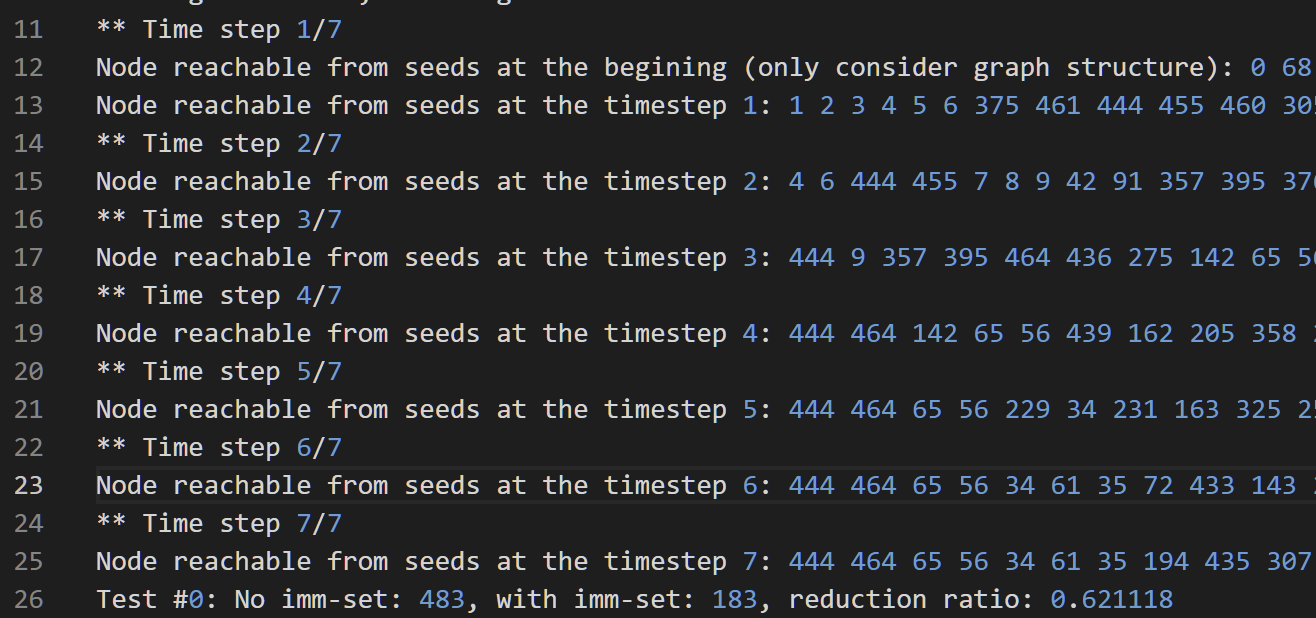
Test #**X**: No imm-set: **A**, with imm-set: **B**, reduction ratio: **C**

其中X表示模拟次数的编号，从0开始到evalcnt-1。而A表示未进行影响力阻挡时，被激活节点的个数。B则表示进行了影响力阻挡后，被激活节点的个数。C表示影响力阻挡降低了被激活节点的比例，即C=(A-B)/A，C精确到六位小数输出。

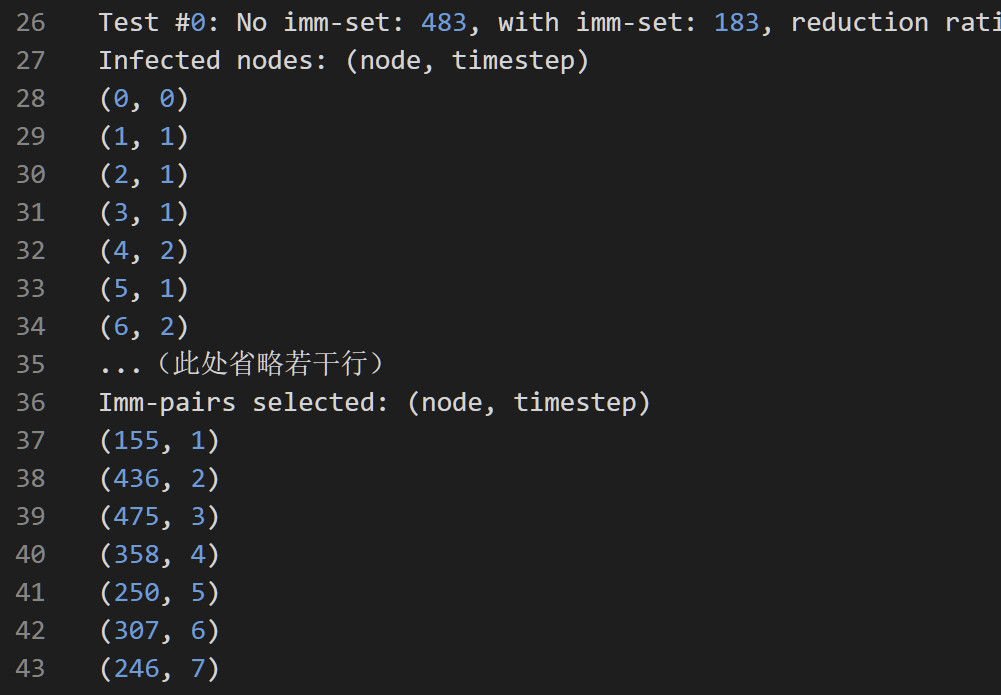
5. 模拟结果输出后的下面一行为程序的运行时间，单位为秒，测量的是CPU时间。

6. 输出文件最后一行为所有的evalcnt次模拟的降低比率（即上面第4项中的C）的平均值。

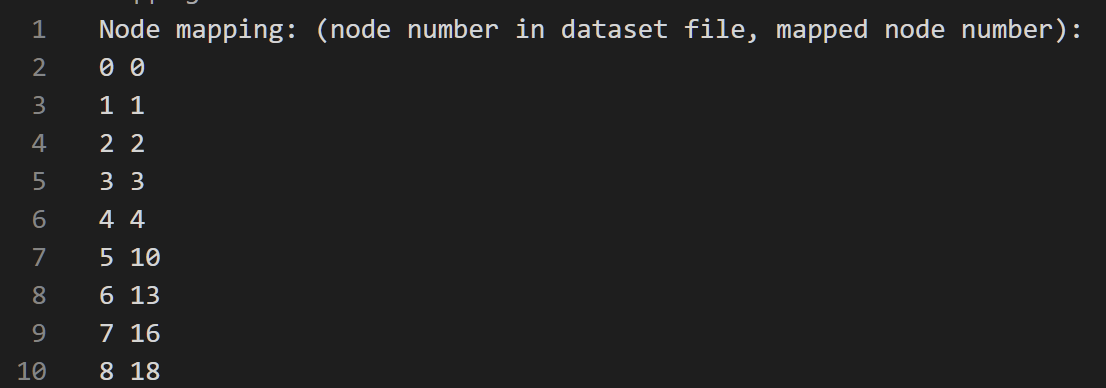
而当verbose=1时，则还会额外输出两个文件，并且在上述.log文件中有额外输出，下面进行介绍。



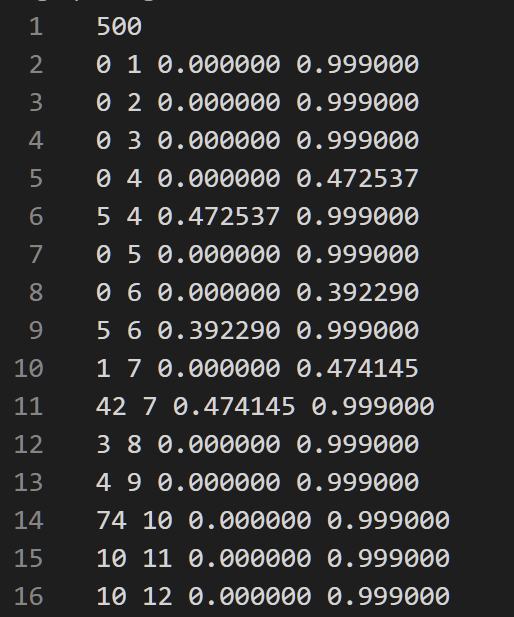
如上图所示，verbose=1时，在每次模拟方式的结果前会输出每个时间步从种子节点可达的节点的标号。其中首个时间步还会输出仅考虑图的结构时可达节点的标号。这些标号全部在一行中输出，上述截图中省略了后面的部分。



如上图所示，verbose=1时还将在每次模拟的结果之后输出被激活的节点以及这些节点被激活的时间步（上图中第27行到第34行），以及选择的阻挡节点和时间步的对（上图中第36行到第43行）。此处的输出格式均为(v, t)形式，其中v为节点，t为对应的时间步。



如上图所示为verbose=1时的额外输出文件之一的样例，该文件名称与配置文件名无关，为nodemapping.out。该文件输出了上文描述的节点重标号的结果。在第二行开始，每行输出了两个用空格隔开的整数a和b，表示原始数据集文件中的标号为a的节点，在软件中被重标号为b。该文件的行数应当为节点数量加一。



如上图所示为verbose=1时的另一个额外输出文件的样例，该文件名称与配置文件名无关，为graphweights.out。该文件输出了软件生成的每条边的权值，其具体生成方法上文已经提及。的输出格式如下：

n

u\_1 v\_1 ev\_1 cv\_1

u\_2 v\_2 ev\_2 cv\_2

...

u\_m v\_m ev\_m cv\_m

其中第一行输出的整数n表示数据集中的节点总数。假设数据集中边数为m，则之后一共有m行，每行包含u, v, ev和cv四个整数。其中u和v分别表示一条边的起点和终点。可以从样例截图中注意到，边的输出顺序是以v，即边的终点排序的，这和上文提到的边权生成方式有关。而ev表示从u到v的这条边的边权，cv则表示当前已输出的所有以v为终点的边的边权累计值。注意到，根据边权生成方式，在正常情况下，当v的所有边都被输出后，cv必然等于参数wsum，这点在截图中也体现了出来，我们可以从截图中推测得知该样例的wsum=0.999。