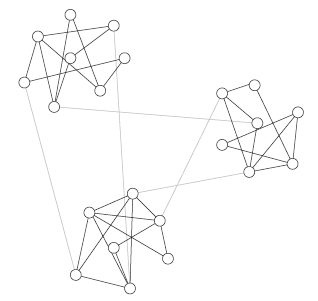
**README.md**

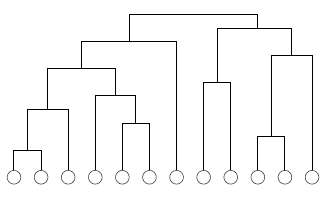
**社区发现**

    先来说说什么是社区发现吧，学术上的说法是：一个社区由一组连接紧密的结点组成，同时这些结点与社区外部的结点连接稀疏，如下图所示。那么，社区发现就是在复杂网络中发现这些连接紧密的社区结构。其实，我个人觉得，社区发现就是网络中的结点聚类。 [](https://camo.githubusercontent.com/e6aa1d5bc67a0077c7062c476267a14f4d67fe19/687474703a2f2f37786e6662632e636f6d312e7a302e676c622e636c6f7564646e2e636f6d2f474e5f312e706e67)

**GN算法**

    GN算法[1]是社区发现中的第一个算法，也就是它开启了这个研究领域。它的基本思想是删除掉那些社区之间的连接，那么剩下的每个连通部分就是一个社区。那么问题来了，就是那些是社区之间的边呢？作者巧妙地借助最短路径解决了这个问题，他们定义一条边的介数(betweeness)为网络中所有结点之间的最短路径中通过这条边的数量，而介数高的边要比介数低的边更可能是社区之间的边。其实，这也比较好理解，因为两个社区中的结点之间的最短路径都要经过那些社区之间的边，所以它们的介数会很高。

    GN算法每次都删除网络中介数最大的边，直至网络中的所有边都被删除。这样GN的过程对应着一颗自顶向下构建的层次树。在层次树中选择一个合适的层次分割即可。

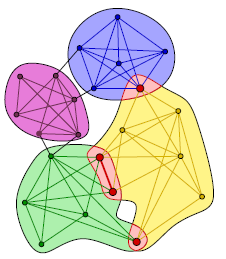
[](https://camo.githubusercontent.com/cd50c96bc7415b19723ce7389ac59cf3ed7e00b0/687474703a2f2f37786e6662632e636f6d312e7a302e676c622e636c6f7564646e2e636f6d2f474e5f322e706e67)

    GN算法的准确性很好，但是时间复杂度却太高，显然找到所有最短路径很耗时。使用[2]中的方法计算介数的时间复杂度为O(mn)，所以GN的时间复杂度为O(m2n)，m是边数量，n是结点数量。

**CPM**

**简介**

    k-团渗透算法(CPM)[3]是第一个能够发现重叠社区的算法，重叠社区指的是结点可以同时属于多个社区。重叠社区在社交网络中是十分常见的，因为每个人都有着多种多样的社交关系。

[](https://camo.githubusercontent.com/432276dc136f8cc73e9264a845ad0b02a7bb2e44/687474703a2f2f37786e6662632e636f6d312e7a302e676c622e636c6f7564646e2e636f6d2f43504d5f312e706e67)

**算法**

    网络中的最大团指的是，团中任意两个结点之间都有边连接，并且它不被其他的团所包含。CPM算法的想法非常简单，首先它找出网络中所有大小至少为k的最大团。然后构建一个团图，每个最大团都是团图中的一个结点，如果两个团c1与c2共享min(c1,c2)-1个邻居的话，它们在新图中的结点之间就存在边。最后团图中的每个连通单元就是一个结点的社区，而它可能是重叠的。

**并行化**

挖掘最大团的过程可以改造为map reduce格式的，详细过程请见[5]

**LPA**

**简介**

    基本的标签传播算法(LPA)[6]的思想非常简单，就是让每个结点与它的大多数邻居在同一个社区中。具体算法流程为：初始化，每个结点携带一个唯一的标签；然后更新结点的标签，令其标签与它的大多数邻居的标签相同，若存在多个则随机选择。迭代直至每个结点的标签不再变化。

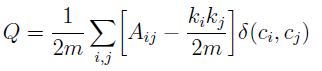
    LPA算法的优点是简单、快速接近线性时间，5次迭代就可使95%的结点标签稳定。缺点是算法结果不稳定，多次执行可能得到的结果都不同。

    针对基本的标签传播算法有时会形成过大("monster")的社区，[7]提出一个令标签跳跃衰减的方法。初始时给每个标签权重为1.0，在更新结点标签时，令其与它的邻居标签中权重最大的相同，并令权重损失一部分。

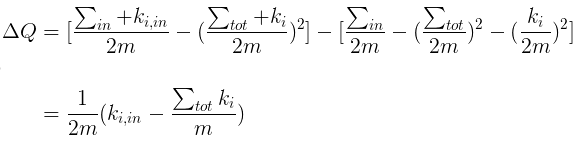
**Louvain**

**简介**

       Louvain算法[8]是一种基于多层次优化Modularity[9]的算法，它的优点是快速、准确，被[10]认为是性能最好的社区发现算法之一。Modularity函数最初被用于衡量社区发现算法结果的质量，它能够刻画发现的社区的紧密程度。那么既然能刻画社区的紧密程度，也就能够被用来当作一个优化函数，即将结点加入它的某个邻居所在的社区中，如果能够提升当前社区结构的modularity。

       Modularity的定义如下：        [](https://camo.githubusercontent.com/366b414ca889a86ca7e64fbf03f268ba2be919f8/687474703a2f2f37786e6662632e636f6d312e7a302e676c622e636c6f7564646e2e636f6d2f4c6f757661696e5f312e706e67)

       其中，m表示网络中边的数量，A为邻接矩阵，如果ci,cj相同则$\delta(ci,cj)$＝1否则为0。

       如果当前结点所在的社区只有它自己，那么在计算将它加入到其它社区时的modularity的变化有个技巧来加速计算，Louvain的高效性也在一定程度上受益于此，它为:[](https://camo.githubusercontent.com/cff8f334c32babb3b4e2d8b97343cecb43027531/687474703a2f2f37786e6662632e636f6d312e7a302e676c622e636c6f7564646e2e636f6d2f4c6f757661696e5f322e706e67)

       Louvain算法包括两个阶段，在步骤一它不断地遍历网络中的结点，尝试将单个结点加入能够使modularity提升最大的社区中，直到所有结点都不再变化。在步骤二，它处理第一阶段的结果，将一个个小的社区归并为一个超结点来重新构造网络，这时边的权重为两个结点内所有原始结点的边权重之和。迭代这两个步骤直至算法稳定。它的执行流程如图所示：        [https://camo.githubusercontent.com/0a912dba35cd5455ad82e4464e571479b19936f0/687474703a2f2f37786e6662632e636f6d312e7a302e676c622e636c6f7564646e2e636f6d2f4c6f757661696e5f332e706e67](https://camo.githubusercontent.com/0a912dba35cd5455ad82e4464e571479b19936f0/687474703a2f2f37786e6662632e636f6d312e7a302e676c622e636c6f7564646e2e636f6d2f4c6f757661696e5f332e706e67)

**代码实现**

GraphX是Spark上的一个图处理框架，它在RDD的基础之上封装出VertexRDD以及EdgeRDD，由这两个封装出的RDD便可构成图结构， [详细请见官网。](https://spark.apache.org/docs/latest/graphx-programming-guide.html)

**参考文献**

1. Community structure in social and biological networks
2. Finding and evaluating community structure in networks
3. Uncovering the overlapping community structure of complex networks in nature and society
4. The worst-case time complexity for generating all maximal cliques and computational experiments
5. Efficient Dense Structure Mining using MapReduce
6. Near linear time algorithm to detect community structures in large-scale networks
7. Towards Real-Time Community Detection in Large Networks
8. Fast unfolding of communities in large networks
9. Finding community structure in very large networks
10. Community detection algorithms: A comparative analysis