算法

本文提供了多种不同算法的比较，根据如何识别社区分为了五类，在后面进行合成网络和真实的社交网络测试的算法有十四个。先对这十四个进行整体分类后面再详细介绍每一种。

- 派系过滤方法（Clique Percolation）CFinder
- 线图和链路分区（Line Graph and Link Partitioning）Link
- 局部扩展和优化（Local Expansion and Optimization）CIS、LFM、OSLOM、UEOC、iLCD、EAGLE、GCE
- 模糊检测（Fuzzy Detection）MOSES、NMF
- 基于代理和动态（Agent-Based and Dynamical）COPRA、SLPA、Game

一、派系过滤方法（Clique Percolation）

派系过滤方法 CPM 基于以下假设：社区由完全连接的子图的重叠集组成，并通过搜索相邻团来检测社区。

描述：定义 k-派系结构是网络中包含 k 个节点的完全子图，所有彼此连通的 k-派系的集合构成一个 k-派系社区。CPM 算法通过对网络中各节点的度可以判断可能存在的最大派系，从一个节点出发找到包含该节点的大小为 k 的派系，采用迭代回归的算法寻找网络中大小不同的派系，最后将彼此连通的 k-派系集合为一个社区结构。

步骤：

Step1：识别网络中大小为 k 的所有团。

Step2：构造一个新的图，使得每个顶点表示为这些 k-派系中的一个。如果表示它们的 k-派系共享 k-1 个成员，则连接两个节点。

下面是一个 4-派系社区的图例：

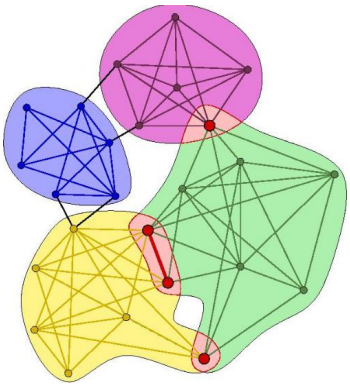


Fig.1. Illustration of the four k-clique communities at k = 4

注意：由于顶点可以同时处于多个 k-派系，所以社区之间可能存在重叠。

评价：CPM 适用于比较密集的网络，而且 k 值较小时结果较好。CFinder 是 CPM 的实现。由于即使小的网络可以包含大量的 k-派系，这种方法的实现是基于定位最大派系而不是个体 k-派系。因此，这种方法的复杂性在实践中有大的时间复杂性（尽管发现 k-派系是多项式），所以不能在大型社交网络中使用。

扩展：

加权派系过滤算法 CPMw[Farkas et al.2007]把子图强度有效地结合到派系搜索中，根据设定的固定强度阈值，将大于此阈值的 k-派系包含到社区中。与 CPM 算法不同的是，CPMw 算法从整体考虑派系强度，因而允许 k-派系中包括低于预设阈值的边，从而得到更稳定的社区划分。注意：子图中链路权重的几何平均值称为该子图的强度。

快速派系过滤算法 SCP[Kumpula et al.2008]该算法根据边的权重递减的顺序将边添加到网络，并记录新出现的 k-派系。再根据得到的 k-派系结构，检测 k-派系之间存在的重叠程度，以判断 k-派系是否已包含在之前 k-派系社区之中，逐步更新以确定最终的社区划分。由于每个 k-派系被处理两次，SCP 算法的计算复杂度与网络中 k-派系数量成线性关系，其运行时间相比于 CPM 明显降低。该算法能有效揭示网络中社区的层次结构，另外在处理加权网络时，CPM 算法需迭代计算每个可能的权重阈值下的社区，而 SCP 算法只需要运行一次就能获得多阈值下的社区结构同时嵌套的社区结构。在处理大规模加权社交网络，当缺乏有关合适阈值的先验知识或研究多阈值下社区结构时，SCP 算法就显得非常有效。

二、线图和链路分区（Line Graph and Link Partitioning）

链路分割算法是分割链接而不是节点来发现社区结构，如果连接到一个节点的链接被放在多个簇中，则这些在原始图中的节点被称为是重叠的。

描述：通过边的相似性聚类划分链接。

步骤：

Step1：给定在节点 k 上的一对链路 e_{ik} 和 e_{jk}，通过 Jaccard 指数来计算相似性 $S(e_{ik}, e_{jk}) = \frac{|N_i \cap N_j|}{|N_i \cup N_j|}$ 其中 N_i 是包括节点 i 的邻域。

Step2：使用单链层次聚类来构造以 k 为根的链接树形图。

Step3：在某个阈值下，切割这个树形图来产生链接社区。

评价：利用链路分割的重叠社区检测不能保证比基于节点的检测有更高的质量，因为这些算法依赖于社区的模糊定义，比如说上述的各种阈值，并不能说明哪个才是最适合的。

扩展：

Evans[Evans and Lambiotte 2009;2010]将网络投影到加权线图中，原来图中的链接成为现在图中的节点，然后就可以应用不相交的社区检测算法。线图的节点分区对应于原始图边的分区。

CDAEO[Wu et al.2010]提供了用于确定重叠程度的后处理过程。当线图初步分割完成，对于每个节点 i 具有$|E_{icmin}|$或$|E_{icmax}|$的预定阈值，其中 $E_{icmin(cmax)}$是社区中具有最小（最大）连接数的边的集合，$|E_{icmin}|$或$|E_{icmax}|$表示边的数目，将线图中识别的 E_{icmin} 集合的链接去除，这实质上将节点 i 减少到成为单个成员，这样能更好的确定社区的重叠程度。

基于链接的扩展模块化[Nicosia et al. 2009]重叠社区的模块度建立在链接的归属系数上。在社区 c 中，节点 i 到节点 j 的一条链接 l(i,j)表示为$\beta_{l(i,j),c} = F(a_{ic}, a_{jc})$，则从 i 到 j 的任何可能的链接 l(i,j)的期望归属系数可以被定义为 $\beta^{out}_{l(i,j),c} = \frac{1}{|V|}\sum_{j \in V} F(a_{ic}, a_{jc})$（其实就是求均值），在社区中指向节点 j 的任何链路 l(i,j)的期望归属系数定义为 $\beta^{in}_{l(i,j),c} = \frac{1}{|V|}\sum_{i \in V} F(a_{ic}, a_{jc})$（它与上一个公式的不同之处在于“链接所指的方向”）。上述归属系数被用作

观察链路 A_{ij} 和在空模型中从 i 到 j 开始的链路的概率$\frac{k_i^{out}k_j^{in}}{m}$的权重，因此得到新的模块化定义为$Q_{ov}^{Ni}=\frac{1}{m}\sum_c\sum_{i,j\in V}\left[\beta_{l(i,j),c}A_{i,j}-\beta_{l(i,j),c}^{out}\beta_{l(i,j),c}^{in}\frac{k_i^{out}k_j^{in}}{m}\right]$其中$k_i^{out(in)}$是节点 i 的输出（输入）链路的数量，m 是边的总数。Q_{ov}^{Ni}取决于链路属性系数 $F(a_{ic},a_{jc})$，其可以是 a_{ic} 和 a_{jc} 的乘积、平均值或最大值。

三、局部扩展和最优化（Local Expansion and Optimization）

利用局部扩展和优化的算法是基于增长的社区或者部分社区，它们中的大多数依赖于密集连接节点组质量的局部效益函数。

描述：Baumes [Baumes et al.2005]提出了“两步走”过程。

步骤：

Step1：初步形成“种子”

根据一些标准用 RankRemoval 算法对节点进行排序。然后，迭代地移除高度排序的节点，直到形成小的不相交的集群核。这些集群核作为第二步的“种子”。

Step2：迭代扫描（IS）

就是通过添加或删除节点来扩展由第一步形成的“种子”，每扩展一次就计算局部密度函数，直到局部密度函数的值不再改变为止。密度函数可以表示为$f(c)=\frac{w_{in}^c}{w_{in}^c+w_{out}^c}$，其中$w_{in}^c$和$w_{out}^c$是社区 c 总的内部和外部的权重。社区检测的质量取决于种子的质量。

评价：由于该算法允许在扩展期间去除顶点，所以迭代扫描（IS）可能会产生部分断开的情况。

扩展：

CIS[Kelley2009]“两步走”的修改版本，在每次迭代之后检查连接性。当产生断开时，只保留密度最大的那个。CIS 还开发了包含边缘概率 e_p 的新的适应度函数$f(c)=\frac{w_{in}^c}{w_{in}^c+w_{out}^c}+\lambda e_p$。参数λ控制算法在网络的稀疏区域中的行为，节点的添加需要在内部密度的变化和边缘密度的变化之间取得平衡。

随机种子节点的局部优化方法 LFM [Lancichinetti et al. 2009]定义了一种节点适应度函数$f(c)=\frac{k_{in}^c}{(k_{in}^c+k_{out}^c)^\alpha}$来度量社区内部边与社区外部边的关系。基于节点适

应度的贪婪优化，从随机种子节点扩展直到适应度$f(c)=\frac{k_{in}^{c}}{(k_{in}^{c}+k_{out}^{c})^{\alpha}}$局部最大来形成社区，其中$k_{in}^{c}$和$k_{out}^{c}$分别是社区 c 内部和外部的总程度，α是控制社区大小的分辨率参数，通过对不同分辨率参数进行调节可得到社区的层次结构。在找到一个社区之后，LFM 随机选择尚未分配给任何社区的另一个节点来生长新社区。注意：由于 LFM 算法的种子节点的选取是随机的，可能导致不稳定的结果。

MONC[Havemann et al. 2011]修正 LFM 的适应度函数为$f(c)=\frac{k_{in}^{c}+1}{(k_{in}^{c}+k_{out}^{c})^{\alpha}}$表示允许单个节点本身被认为是社区。这就避免了违反局部性原则，使得 MONC 能够找到其中一组节点是局部最优的α（分辨率参数）的范围。MONC 计算α的下一个最低值来进一步扩展社区。当一个节点 i 形成的社区是另一节点形成社区的子集时，i 的分析停止（因为如果继续扩展，必然还是形成原来已知的那个社区，现在停止就可以减少以后的计算量）。因为 MONC 在处理期间合并社区，所以与 LFM 相比能更快地揭示网络。

OSLOM[Lancichinetti et al. 2011]关于全局空模型（由配置模型生成的随机图）在社区扩展期间，测试了一个簇的统计数据。为了扩展当前社区，针对每个邻居计算 r 值（在空模型中，内部连接数≧从邻居到该社区的连接数的累积概率），如果最小 r 值的累积概率小于给定容差，则认为是有意义的，将相应的节点添加到社区。否则，检查第二小的 r，以此类推。OSLOM 通常导致大量的异常值或单例社区。

UEOC [Jin et al. 2011]基于相应退火算法，采用随机游走的 l 步转移概率扩展节点形成社区。按转移概率降序排序节点之后，以某种适当的截止值提取社区。模拟退火算法以一定的概率来接受一个比当前解要差的解，因此有可能会跳出这个局部的最优解，达到全局的最优解。

OCA [Padrol-Sureda et al. 2010]基于将每个节点映射到 d 维向量的思想。节点 S 的每个子集被定义为该集合中的各个向量的和。适应度函数被定义为函数 O 上的有向拉普拉斯算子，其中 O 是子集向量的平方欧几里德长度。像 LFM 一样，从一些初始种子开始，OCA 试图去除或添加导致适应度函数的值最大增加的节点。

iLCD[Cazabet et al. 2010]能够检测静态和暂时的社区。给定创建的一组边，

如果其第二邻居的数目和鲁棒第二邻居的数目大于期望值，则 iLCD 通过添加新节点来更新现有社区。如果检测到的是最小模式，则还允许由新边创建新社区。将两个社区之间的相似性定义为共同节点的比率，如果相似性高，则执行“合并”来提高检测效率。

EAGLE[Shen et al. 2009]使用聚集框架来产生树形图。

Step1：发现所有最大派系并且使其成为初始社区，派系对于许多局部优化算法是非常重要的。

Step2：合并具有最大相似度的一对社区。

GCE[Lee et al. 2010]第一步与 EAGLE 相同。

Step1：将最大派系识别为初始社区。

Step2：通过贪心算法优化局部适应函数来扩展社区，$1-\frac{|c_1\cap c_2|}{min(|c_1|,|c_2|)}$ 定义为的社区 c_1 和 c_2 之间的距离，如果 c_1 与 c_2 类似(也就是说 $1-\frac{|c_1\cap c_2|}{min(|c_1|,|c_2|)}$ 小于给定参数ϱ)，则去除与先前发现类似的社区。

四、模糊检测（Fuzzy Detection）

模糊检测算法量化所有节点和社区对之间的关联强度。在这些算法中，为每个节点计算成员隶属向量或归因因子。缺点是需要确定向量的维数 k。

Nepusz [Nepusz et al. 2008]将重叠社区检测建模为非线性约束优化问题，可以通过模拟退火的方法来解决。目标是要最小化函数 $f=\sum_{i=1}^{n}\sum_{j=1}^{n}w_{ij}(\tilde{s_{ij}}-s_{ij})^2$ 的值，w_{ij} 是预定义的权重，$\tilde{s_{ij}}$ 是节点 i 和 j 之间的已知的相似度，相似度 s_{ij} 定义为 $s_{ij}=\sum_{c}a_{ic}a_{jc}$，$a_{ic}$ 是社区 c 与节点 i 的模糊隶属度，满足 $0\le a_{ic}\le 1,\forall i\in V,\forall c\in C$ 且 $\sum_{c=1}^{|C|}a_{ic}=1$（f 表示的是已知的相似度与实际测得的相似度的差距）。节点 i 的归属因子被定义为 $Q_{ov}^{Nc}=\frac{1}{2m}\sum_{c}\sum_{i,j\in c}\left[A_{ij}-\frac{k_ik_j}{2m}\right]a_{ic}a_{jc}$，$a_{ic}$ 是节点 i 在社区 c 中的隶属程度，增加 k 的值，直到社区结构没有改善，可以确定社区 k 的值。

SSDE[Magdon-ismail and Purnell 2011]类似的混合模型被构造为节点的生成模型。

Step1：首先使用频谱聚类方法将网络映射到 d 维空间。

Step2：通过预期最大化算法训练高斯混合模型(GMM)。

Step3：当添加簇的对数似然性的增加不明显高于将簇添加到在同一空间上均匀的随机数据时，确定社区的数量。

非负矩阵因子分解（NMF）是经常在社区检测的机器学习中进行特征提取和维数降低的技术。NMF 将特征矩阵 V 近似因子分解为具有非负性约束的两个矩阵 V ≈WH，其中 V 是 n×m，W 是 n×k，H 是 k×m，k 是用户提供的社区数。W 表示缩减特征空间中的数据，节点 i 相对于社区 j 的依赖性量化为归一化的 W 中的每个元素 $w_{i,j}$ 值。[Zhang et al. 2007b]把特征矩阵 V 替换为扩散核，扩散核是网络中拉普拉斯算子的函数。[Zarei et al. 2009]特征矩阵 V 被定义为拉普拉斯的列的相关矩阵，使得比[Zhang et al. 2007b]有较好的性能。[Zhao et al. 2010]去除了近似的冗余约束，将 NMF 简化为对称非负矩阵分解(s-NMF)的问题。

Wang et al[Wang et al. 2009]使用局部优化算法与不相交检测方法的组合。

Step1：利用任意的不相交团体检测算法获得初始社区。

Step2：社区尝试添加或删除节点。

Step3：计算包括节点 i 或去除节点 i 的社区上的两个适合度分数的方差。

Step4：归一化方差形成节点 i 的模糊隶属向量。

Ding [Ding et al. 2010]采用相似性传播聚类算法进行重叠检测，其中簇由代表性样本识别。

Step1：通过通勤时间内核（逆拉普拉斯算子的函数）将节点映射为欧几里德空间中的数据点。

Step2：通过余弦距离来测量节点之间的相似性，由此相似性来确定社区结构。

五、基于代理和动态算法（Agent-Based and Dynamical Algorithms）

标签传播算法[Raghavan et al. 2007; Xie and Szymanski 2011]其中具有相同标签的节点形成社区，通过允许节点具有多个标签可以扩展到重叠的社区检测上。

COPRA[Gregory 2010]每个节点通过以同步方式，在每个时间步长，对来自其所有邻居的系数进行求平均值来更新其归属系数。

SLPA[Xie et al. 2011; Xie and Szymanski 2012]是基于通用讲话者的信息传播过程。它根据成对的交互规则在节点之间扩展标签，节点忘记在先前迭代中获得

的信息，SLPA 向每个节点提供存储器以存储接收到的信息（标签）。在节点的存储器中观察标签的概率称为隶属强度。SLPA 不需要先前知道社区的数量，这是由网络中标签的聚类确定的。SLPA 还可以通过泛化交互规则（称为 SLPAw）来应用于加权和定向网络。

游戏理论框架[Chen et al. 2010]社区与纳什局部均衡有关。增益函数和损失函数与每个代理相关联。Game 假定每个代理都是自私的，它们根据自己的效用选择加入、离开和切换社区。只要能增加效用，就允许代理联合多个社区来处理重叠。