社区检测及强连通图计算的方法研究

窦洪健 付岩松 王皓 王帅

2016100851 2016103559 2016103561 2016100861

摘 要 近年来,对于复杂网络的研究受到了广泛关注。现实世界中的许多系统,如社会关系网、互联网、生物网络、交通网络等等都可以建模成为复杂网络,用网络中的节点表示现实世界中的各个事物,而事物间的联系用网络中节点间的连接表示。社区结构是网络的重要特征之一,它表征着该网络中节点的不同划分。归于同一个网络社区的节点间联系较紧密,而不同社区中的节点之间联系较为稀疏。检测出网络中的社区,有助于我们分析该网络中各个事物间的关系,把握网络的整体结构。因此,对于网络社区检测方法的研究是一个非常有意义的课题。现有的很多网络社区检测算法需要预先设定社区数目,如基于谱方法的社区检测算法,而实际问题中,我们往往未知网络的社区数目,这就要求我们提出可以自动检测出所给网络中社区数目的方法。本文利用多种算法对重叠及非重叠社区进行划分及可视化，分析出不同算法的优缺点。此外，有向图的强连通分量应用非常广泛，深度优先遍历是求有向图的强连通分量的一个有效方法，本文计算出强连通图的数量及相应的直径。

关键词 社区检测；强连通图；直径；Louvain 算法；图

Research on the Method of Community Detection and Strong Connected Graph Calculation

Hongjian Dou Yansong Fu Wanghao Wangshuai

Abstract In recent years, the study of complex networks has received extensive attention. Many systems in the world, such as social networks, the Internet, biological networks, transport networks, and so on, can be modeled as complex networks. Nodes in the network can be represented everything in the real world, and the links between things are represented by the connection of nodes in the network. Community structure is one of the important characteristics of the network, which characterizes the different divisions of nodes in the network. The links between nodes belonging to the same network community are closer, and the links between nodes in different communities are relatively sparse. Detection of the community in the network help us to analyze the relationship between the various things in the network, grasp the overall structure of the network. Therefore, the study of online community detection method is a very interesting subject. Many of the existing online community detection algorithms need to set the number of communities in advance, such as the community detection algorithm based on the spectral method. But in fact, we don’t know the number of network communities in many cases, which requires us to give some methods of detecting the number of network in the community automatically. In this paper, we use multiple algorithms to divide and visualize overlapping and non-overlapping communities, and analyze the advantages and disadvantages of different algorithms. In addition, the strongly connected component of the directed graph is very widely used. The depth-first traversal is an effective method to find the strong connected components of the directed graph. In this paper, the number of strong connected graphs and the corresponding diameters are calculated.

Key words Community detection; strongly connected components; diameter; Louvain algorithm; Graph

1 非重叠社区的检测

用复杂网络来对各个领域中的系统进行建模，网络中的节点表示系统中的事物，网络中连接各个节点的边，表示各个事物之间的关系。网络中的社区结构[1]是网络的重要特性之一，社区结构反映了网络中各个节点间的关系及网络的整体结构。同一个社区中的节点，它们之间有较大的相似性，联系较为紧密而与网络中的其他部分的节点联系相对而言比较稀疏。也就是说，在社区内部，节点之间的联系非常紧密，而社区之间的联系相对而言比较稀疏。

社区结构揭示了给定网络隐藏的特性。因此，对网络中社区的检测不仅是研究网络中实体是如何组合在一起的基本步骤，也是更好理解大规模网络全局结构和功能特性很好的方法[2]。

复杂网络中的社区检测给人们提供信息以便更好了解网络内部成员及他们之间的关系，因此，社区检测问题已受到广泛关注，解决该问题的许多方法已被提出。现有网络社区检测方法可分为两大类:基于优化的和启发式算法[3] 。在基于优化的算法中，社区检测问题被定义为对表征网络划分优劣的目标函数的优化问题。启发式算法将问题定义为对启发式规则的设计。

目前，对于非重叠社区检测的算法主要有基于模块度优化的社团发现算法（GN）、基于标号传播的社团发现算法（LPA）[4]、Louvain算法[5]、基于谱分析的社团发现算法和基于信息论的社团发现算法。下面主要介绍下前三种算法。

1.1 GN算法

GN 算法的基本思想是：在一个网络之中，通过社区内部的边的最短路径相对较少，而通过社区之间的边的最短路径的数目则相对较多。GN 算法是一个基于删除边的算法，本质是基于聚类中的分裂思想，在原理上是使用边介数（边介数指网络中任意两个节点通过此边的最短路径的数目）作为相似度的度量方法。在 GN 算法中，每次都会选择边介数高的边删除，进而网络分裂速度远快于随机删除边时的网络分裂。

GN 算法的步骤如下：

（1）计算每一条边的边介数；

（2）删除边介数最大的边；

（3）重新计算网络中剩下的边的边介数；

（4）重复(3)和(4)步骤，直到网络中的任一顶点作为一个社区为止。

GN 算法的缺陷：

（1）不知道最后会有多少个社区；

（2）在计算边介数的时候可能会有很对重复计算最短路径的情况，时间复杂度太高；

（3）GN 算法不能判断算法终止位置。

1.2 LPA算法

LPA 算法的基本思想是：通过标记节点的标签信息预测未标记节点的标签情况。节点之间的标签传播主要依照标签相似度进行，在传播过程中，未标记的节点根据邻接点的标签情况来迭代更新自身的标签信息，如果其邻接点与其相似度越相近，则表示对其所标注的影响权值就越大，邻接点的标签就更容易进行传播。起初每个节点拥有独立的标签，那么网络中有 n 不同标签，每次迭代中对于每个节点将其标签更改为其邻接点中出现次数最多的标签，如果这样的标签有多个，则随机选择一个。通过迭代，直到每个节点的标签与其邻接点中出现次数最多的标签相同，则达到稳定状态，算法结束。此时具有相同标签的节点即属于同一个社区。

LPA 算法优缺点：在该算法中，初始的标签数据就像是一个源头，按照一定规则对相邻节点进行标注，标签沿着网路连接进行传播，节点间相似度越大，标签越容易传播。LPA 该算法简单易实现，算法执行时间短，复杂度低，缺点是每次迭代结果不稳定，准确率不高。

1.3 Louvain算法

Louvain 算法是基于模块度的社区发现算法，该算法在效率和效果上都表现较好，并且能够发现层次性的社区结构，其优化目标是最大化整个社区网络的模块度。模块度（也称 Q 值）用于度量网络中各社区内部联系的强度，在一个高 Q 值的网络中，各个社区内部的链路较为密集而社区间的链路则十分稀疏。因此，Q 值可以作为一个优化问题的目标函数，求得其（近似的）全局最优解，即可得到这个网络的最优的社区划分。它的物理含义是社区内节点的连边数与随机情况下的边数只差，它的取值范围是 $[-1/2,1)$，其定义如下：

$$Q = \frac{1}{2m}\sum_{i,j}\left[A_{i,j} - \frac{k_i k_j}{2m}\right]\delta(c_i, c_j)$$

其中，$A_{i,j}$代表节点 i 和节点 j 之间边的权重，网络不是带权图时，所有边的权重可以看做是 1；$k_i = \sum_j A_{i,j}$表示所有与节点 i 相连的边的权重之和（度数）；c_i表示节点 i 所属的社区；$m = \frac{1}{2}\sum_{i,j} A_{i,j}$表示所有边的权重之和（边的数目）;而$\delta(c_i, c_j)$函数中两个变量相同时取值为 1，反之为 0。

Luovain 算法步骤如下：

（1）将图中的每个节点看成一个独立的社区，次数社区的数目与节点个数相同；

（2）对每个节点 i，依次尝试把节点 i 分配到其每个邻居节点所在的社区，计算分配前与分配后的模块度变化ΔQ，并记录ΔQ最大的那个邻居节点，如果maxΔQ >0，则把节点 i 分配ΔQ最大的那个邻居节点所在的社区，否则保持不变；

$$\Delta Q = \left[\frac{\sum in + k_{i,in}}{2m} - \left(\frac{\sum tot + k_i}{2m}\right)^2\right] - \left[\frac{\sum in}{2m} - \left(\frac{\sum tot}{2m}\right)^2 - \left(\frac{k_i}{2m}\right)^2\right]$$

（3）重复步骤 2，直到 Q 值不再发生变化，即将一个节点转移到网络内的另一个相邻社区，将不能带来ΔQ的提升，此时当前网络内所有节点都不再移动；

（4）社区归并，这一步也可看做对原图的压缩，将前几步得到的各个社区作为新图的节点，同时将原社区内部所有节点对的边权重之和作为新的权重赋予新图的各条边。

以上步骤其实包含了两个阶段：1-3 步是对 Q 值最优解的求解，第 4 步将这一轮划分得到的社区进行归并，得到一张新图。完成以上两个阶段称为一轮（pass），完成一轮的计算后算法会自动进入下一轮计算的第一阶段，迭代若干轮候会发现最终得到网络的 Q 值不再增长，此时的网络已经聚合为几个内部联系紧密，外部关联稀疏的小社区，此时算法完成。

Louvain 算法有如下优势：

（1）算法得到的社区结构是分层的，每一轮计算完成后得到的新图都是对一个大社区内若干细分社区发现的结果，这样的分层结构是每个网络的自然属性，能深入了解某个社区内部结构和形成机制；

（2）算法易于实现，并且计算过程全程无监督，即最终结果完全依赖于算法聚类，并不需要人为提前预设分类；

（3）算法的性能较好，在进行一些经典社区分类算法的对比中，Louvain 算法对图的大小几乎没有上限要求，并且能在迭代几轮后快速收敛。这为处理拥有百万级别以上节点的移动通信网络甚至上亿节点的大型社交网络的社区发现提供了可能。

1.4 实验结果

本文利用 GN 算法、LPA 算法和 Louvain 算法对社区进行划分。但是由于 GN 算法时间复杂度比较高，没有得到该实验的最终结果。因此，以下将只列出 Louvain 和 LPA 算法的实验结果。

首先来看 LPA 算法的实验结果：

（1） LPA 算法总共划分出近 1200 个社区，下图为其中节点数为 top50 的社区划分结果。

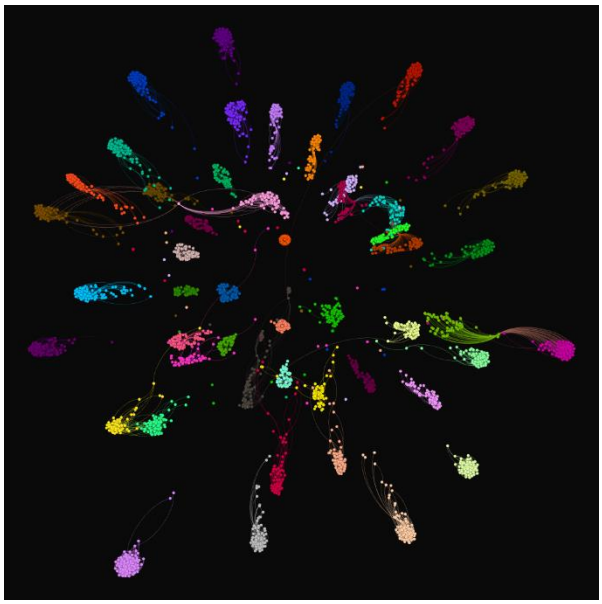


图 1.1 LPA_50 社区

因为 LPA 算法划分的社区数量较多，每个社区内的节点数量很少，因此我们最终只选择展示数量为 50 的社区，没有列出更多或者更少的社区进行展示。

接着来看下 Louvain 算法的实验效果：

（1） Louvain 算法总共划分为 50 个社区，首先我们选取节点数量为 top15 的社区进行展示。下面两张图分别是 gephi 软件 layout 为 Force Atlas 和 Fruchterman Reingold 所绘的图。

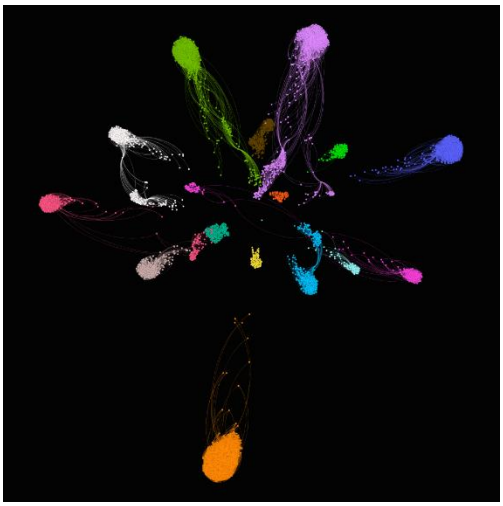


图 1.2 Louvain15_Force Atlas

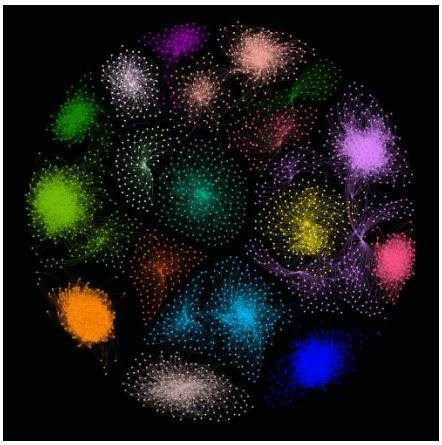


图 1.3 Louvain15_Fruchterman Reingold

（2） 其次，我们制作出节点数量为 top30 社区划分图。由于内存的限制，我们采用 layout 为 Force Atlas2。

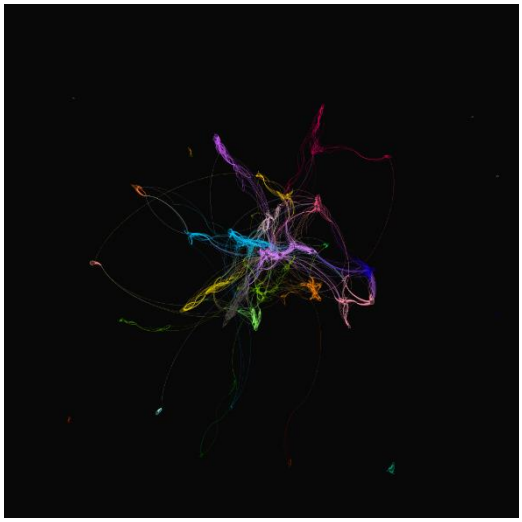


图 1.4 Louvain30_ Force Atlas2

1.5 实验结果分析

根据 LPA 和 Louvain 算法得出的社区划分结果可知，LPA 算法时间复杂度较低，但是精度较低，从实验结果来看，LPA 划分出近 1200 个社区，很多社区中节点数在 5 以内，划分的较为分散。而对于 Louvain 算法来讲，该算法时间复杂度相比较 LPA 较高，但是精度比较高，该算法存在一定问题，主要体现在划分的社区较大，很多并不属于同一社区的节点被划分到同一社区，这是因为在该算法中，节点数量越多，边就越多，相应的 modularity 也就越大，更会被划分到同一社区内。

为了解决 Louvain 算法存在的问题，我们采用了一个比较简单的方法解决，即在首次划分的结果上，我们再次调用 Louvain 算法对单一社区进行划分。实验结果如下图：

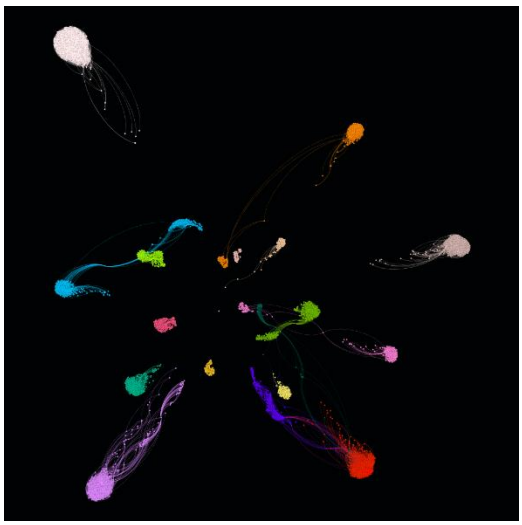


图 1.5 Louvain15_ Force Atlas_update

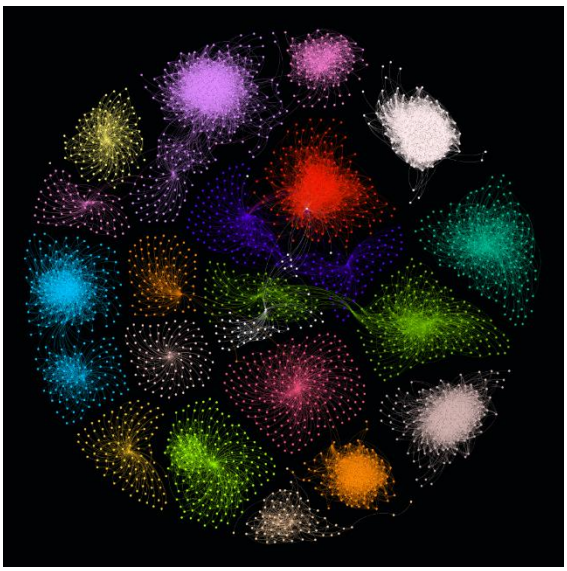


图 1.6 Louvain15_ Fruchterman Reingold_update

根据实验对比结果，图 1.3 中的粉色社团被划分为图 1.6 中的蓝色、白色和红色社团。可以证明我们的处理方式对于拆分大社团是具有一定效果的。

2 重叠社区的检测

传统的社区挖掘算法是将网络划分为若干个独立的社区，每个节点只隶属于一个社区，代表性算法有基于划分的挖掘算法[6]、基于模块性优化的算法[7]、基于标签传播的算法[8]和基于信息论的算法[9]等。最近人们发现，社会网络中的社区之间并不是独立的，即一些节点可以同时隶属于多个社区，例如实际的人际关系中，每个人可以属于家庭、单位、朋友圈等不同的社会团体。因此，社会网络中的重叠社区结构更符合现实。

目前，社会网络中重叠社区的挖掘已经引起越来越多的关注，代表性的算法有 Palla 等提出的团渗算法（clique percolation method, CPM），其通过团过滤算法来发现网络中的社区结构[10]。其次，还有基于种子扩散思想的重叠社区发现算法，此类算法的基本思想是以具有某种特征的子网络为种子，通过合并、扩展等操作向邻接节点扩展，直至获得评价函数最大的社团。Lancichinetti 等提出以若干个节点为种子，通过扩展形成对整个网络的覆盖，即 LMF 算法 (Detecting the Overlapping and Hierarchical Community Structure in Complex Networks)。本文主要利用了 CPM 算法对重叠社区进行检测。

2.1 CPM算法

首先来看派系的定义：CPM 方法(clique percolation method)用于发现重叠社区，其中派系(clique)是任意两点都相连的顶点的集合，即完全子图。

CPM 算法思想：在社区内部节点之间连接密切，边密度高，容易形成派系。因此，社区内部的边有较大可能形成大的完全子图，而社区之间的边却几乎不可能形成较大的完全子图，从而可以通过找出网络中的派系来发现社区。k-派系表示网络中含有 k 个节点的完全子图，如果一个 k-派系与另一个 k-派系有 k-1 个节点重叠，则这两个 k-派系是连通的。由所有彼此连通的 k-派系构成的集合就是一个 k-派系社区。

网络中会存在一些节点同时属于多个 k-派系，但是它们所属的这些 k-派系可能不相邻，它们所属的多个 k-派系之间公共的节点数不足 k-1 个。这些节点同属的多个 k-派系不是相互连通的，导致这几个 k-派系不属于同一个 k-派系社区，因此这些节点最终可以属于多个不同的社区，从而发现社区的重叠结构。

CPM 算法步骤：

（1）寻找网络中极大的完全子图（maximal-cliques），然后利用这些完全子图来寻找 k-派系的连通子图（即 k-派系社区），不同的 k 值对应不同的社区结构。

（2）找到所有的 k-派系之后，可以建立这些派系的重叠矩阵(clique overlap matrix)。在这个对称的矩阵中，每一行（列）代表了一个派系，矩阵的第 i 行第 j 列表示第 i 个团和第 j 个团的公共节点数。所以矩阵中的非对角线元素代表两个连通派系中共享的结点的数目。对角线元素代表派系的规模。

（3）给定参数 k，将重叠矩阵中非对角线上元素小于 k-1，且对角线上元素小于 k 的所有项置 0，其他的元素为 1；这样，所有对角线为 1 的团为 k 团，而非对角线为 1 的团 i、团 j 是相邻的。通过这个矩阵可以得到相应的社区。

（4）找出属于多个不同社区的相同节点，从而发现重叠社区。

CPM 算法缺陷：由于 k 是个输入参数值，从而 k 的取值将会影响 CPM 算法的最终社区发现结果，当 k 取值越小社区将会越大，且社区结构为稀疏。然而，由于该算法是基于完全子图，因此比较适用于完全子图比较多的网络，即边密集的网络，对于稀疏网络效率将会很低，且该算法还无法分配完全子图外的顶点。

2.2 实验结果

本文利用 CPM 算法对重叠社区进行划分。但是由于 CPM 算法时间复杂度较高，多社区重叠的可视化有难度，因此本实验首先利用常规的非重叠社区检测算法构建社团集合，然后对不同粒度的社团集合进行 CPM 算法检测。实验结果如下：

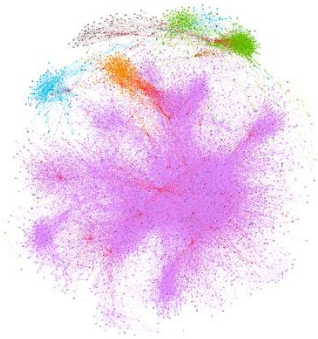


图 2.1 CPM_ Fruchterman Reingold

其中红色的节点表示社区之间重叠的部分，上幅图的实验效果可能不是很清楚，下面对社区进行分解：

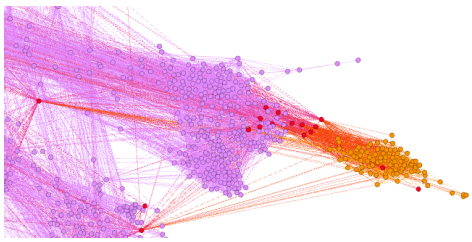


图 2.2 CPM_ Force Atlas2

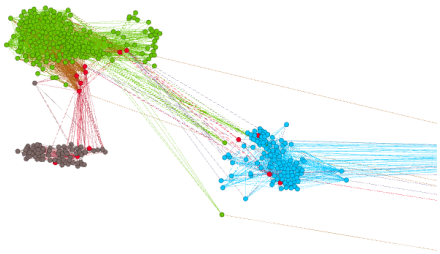


图 2.3 CPM_ Force Atlas2

从图中可知，社区重叠部分大多出现在社区的边缘或者边界部分。

2.3 实验结果分析

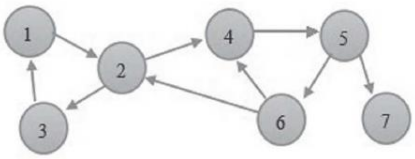
本文利用 CPM 算法对重叠社区的检测其实效果并不是很理想，主要问题存在于以下几个方面。首先，利用 CPM 算法划分的社区规模差异较大，上述实验结果选择了节点个数超过 10 的社区进行展示，CPM 划分的时候还存在大量的小社区，此外，CPM 算法还将部分社区划分的很大，导致很多重叠的节点出现在社区内部，其实这些节点本应该是大社区划分为小社区的边缘。另外，由于时间复杂度较高，本文没有选择全部节点进行检测，一定程度上影响了实验结果。但从实验结果来看，大部分重叠的节点都分布于多个社区的边缘，分布于社区内部的节点也大都是大社区尚未划分为小社区的边缘，从这两方面来看，表明了实验结果的准确性。

3 强连通图计算

在对无向图进行遍历时，对于连通图，仅需一次调用搜索过程。即从图中任一顶点出发，便可遍历访问图中各个顶点。对于非连通图，则需多次调用搜索过程。而每次调用得到的顶点访问序列恰为其各个连通分量中的顶点集。

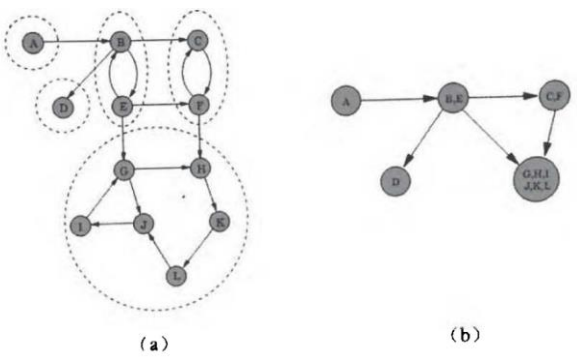
在有向图G中，如果两个顶点V_i，V_j间有一条从V_i到V_j的有向路径，同时还有一条从V_i到V_j的有向路径，则称两个顶点强连通(strongly connected)。如果有向图G的每两个顶点都强连通，称G是一个强连通图。有向图的极大强连通子图，成为强连通分量SCC(Strongly Connected Components)。如果图中一个顶点没有和其它顶点强连通，这个顶点自己也是一个强连通分量。

下图中，子图{1,2,3,4,5,6}和{7}分别是一个强连通分量，因为这个连通分量顶点两两可达。



直接根据定义，有向图的强连通分量不会像无向图的连通分量那样简单。下面看下具体求解过程。

对于下面(a)图我们借助DFS。如果从节点I开始DFS，将得到只包含{G,H,I,J,K,L}的一棵DFS树，然后从节点C出发，得到{C,F}，再从D出发得到{D}，以此类推，每次得到一个SCC。如下面(b)图所示。



但是并不是每个顺序都是有效的。如果一开始选择从节点A开始遍历，这颗DFS树将包含整个图。也就是说，这次DFS遍历把所有SCC混在了一起，什么也没得到。很明显，我们希望按SCC图拓扑顺序的逆序进行遍历。这样才能每次DFS只得到一个SCC，而不会把两个或多个SCC混在一起。Kosaraju算法便是基于这个思想。

深度优先遍历是求有向图的强连通分量的一个有效方法，根据实现方式的不同，总体上，求有向图的强连通分量有三种算法，分别是Kosaraju算法、Gabow算法和Tarjan算法，时间复杂度均为O(n+e)。其中Kosaraju算法要对原图和逆图都进行一次DFS，另外两种算法只要DFS一次，Gabow算法是Tarjan算法的改进。

3.1 基本算法

3.1.1 Kosaraju 算法

Kosaraju 算法基于以下思想：强连通分量一定是某种 DFS 形成的 DFS 树森林。该算法可以说是最容易理解，最通用的算法，它关键的部分是同时应用了原图 G 和 GT。步骤一：先对原图 G 进行深度优先搜索形成森林，步骤二：任选一棵树对其进行深度优先搜素（本次搜索节点 A 能往子节点 B 走的要求是A → B存在于反图 GT），能遍历到的顶点就是一个强连通分量。余下部分和原来的森林组成一个新的森林，继续步骤二直到没有顶点为止。

具体求解步骤如下：

（1） 在有向图中，从各节点出发进行深度优先遍历，并按其所有邻接点的访问都完成（即出栈）的顺序将顶点排列起来。

（2） 在该有向图中，从最后完成访问的顶点出发，沿着以该顶点为头的弧作逆向的深度优先遍历， 若此次遍历不能访问到有向图中所有顶点，则从余下的顶点中最后完成访问的那个顶点出发，继续作逆向的深度优先遍历，依次类推，直至有向图中所有顶点都被访问到为止。

（3） 每一次逆向深度优先遍历所访问到的顶点集便是该有向图的一个强连通分量的顶点集， 若仅作一次逆向深度优先遍历就能访问到图的所有顶点，则该有向图是强连通图。Kosaraju 是对有向图及其逆图两次 DFS 的方法，其时间复杂度也是 O(n+e)。与 Trajan 算法相比，Kosaraju 算法可能会更直观些。但是 Tarjan 只用对原图进行一次 DFS，不用建立逆图。在实际测试中，Tarjan 算法的运行效率也比 Kosaraju 算法高 30%左右。Kosaraju 虽然是线性的但是需要两次 DFS。但是 Kosaraju 算法在计算连通图时同时提供一个计算有向图拓扑排序的线性算法。

3.1.2 Tarjan 算法

Tarjan 算法的主要思想是：强连通分量是 DFS 树中的子树。

搜索时，任选一结点开始进行深度优先搜索（若深度优先搜索结束后仍有未访问的结点， 则再从中任选一点再次进行）。搜索过程中已访问的结点不再访问。搜索树的若干子树构成了图的强连通分量。把当前搜索树中未处理的结点加入一个堆栈，回溯时可以判断栈顶到栈中的结点是否为一个强连通分量。定义 DFN(u)为结点 u 搜索的次序编号(时间戳)，LOW(u)为 u 或 u 的子树能够追溯到的最早的栈中结点的次序号。由定义可以得出，当 DFN(u)=LOW(u)时，以 u 为根的搜索子树上所有结点是一个强连通分量。

算法伪代码如下：

```
tarjan(u)
{
DFN[u]=Low[u]=++Index//为节点 u 设定次序编号和 Low 初值
Stack.push(u)//将节点 u 压入栈中
for each(u,v) in E//枚举每一条边
 if (visnotvisted)//如果节点 v 未被访问过
   tarjan(v)//继续向下找
   Low[u]=min(Low[u],Low[v])
 else if (vinS)//如果节点 v 还在栈内
  Low[u]=min(Low[u],DFN[v])
if (DFN[u]==Low[u])//如果节点 u 是强连通分量的根
repeat{
 v=S.pop//将 v 退栈，为该强连通分量中一个顶点
 printv
```

```
    until(u==v)
  }
}
```

可以发现，运行 Tarjan 算法的过程中，每个节点都被访问了一次，只进出一次堆栈，每条边也只被访问了一次，所以该算法的时间复杂度为 O(n+e)（n 为顶点数，e 为边数）。而 Gabow 算法是 Tarjan 算法的改进。

3.2 计算强连通图数量

由于 Kosaraju 算法实现比较简单，本文采用 Kosaraju 算法计算有向图的强连通图数量。

表 1　连通图数量

连通图个数	单节点连通图	非单节点连通图
12248	11902	346

实验表明，在该网络中强连通图的数量超过 10000 个，但是单节点的连通图占据大部分。

3.3 计算强连通图的直径

在图 G 中 d(u,v)，定义为图中顶点 u 到顶点 v 的一条最短路径。如果没有路径 d(u,v)定义为无穷大。

直径：定义为 max d(u,v)，其中 u,v 是两个顶点。也就是图中距离最远的两个点。

强连通图直径的计算可以归结为计算图中两点之间的最短路径。求解图中两点之间最短路径的方法主要有四种。深度或者广度优先搜索算法，弗洛伊德算法，迪杰斯特拉算法，Bellman-Ford 算法。

（1）　深度或广度优先搜索算法。从起点开始访问所有的深度遍历路径或广度优先路径，则到达终点结点的路径有多条，取其中路径最短的一条则为最短路径。

（2）　弗洛伊德算法。基本思想：最开始只允许经过 1 号顶点进行中转，接下来只允许经过 1 号和 2 号顶点进行中转.....允许经过 1-n 号所有顶点进行中转，来不断动态更新任意两点之间的最短路程。即求从 i 号顶点到 j 号顶点只经过前 k 号点的最短路程。

（3）　迪杰斯特拉算法。基本思想：每次找到离源点（如 1 号结点）最近的一个顶点，然后以该顶点为中心进行扩展，最终得到源点到其余所有点的最短路径。

（4）　Bell-Ford 算法。主要思想：对所有的边进行 n-1 轮松弛操作，因为在一个含有 n 个顶点的图中，任意两点之间的最短路径最多包含 n-1 边。换句话说，第 1 轮在对所有的边进行松弛后，得到的是从 1 号顶点只能经过一条边到达其余各定点的最短路径长度。第 2 轮在对所有的边进行松弛后，得到的是从 1 号顶点只能经过两条边到达其余各定点的最短路径长度。以此类推。

本文考虑到时间复杂度及空间复杂度因素，采用深度优先搜索的方法计算强连通图的直径。实验结果如下：

表 2　直径数量统计

直径大小	1	2	3	4	5	6	7
连通图的数量	292	43	5	2	1	2	1

下面具体展示下直径范围 3-6 的连通图。

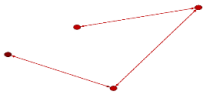


图 3.1　直径为 3 的连通图

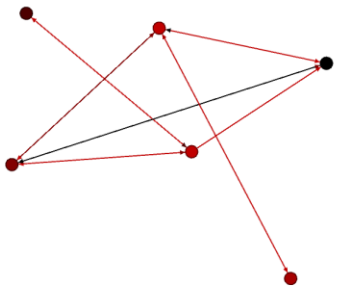


图 3.2　直径为 4 的连通图

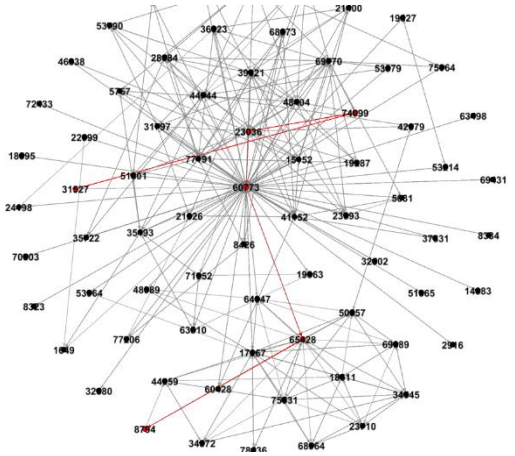


图 3.3 直径为 5 的连通图

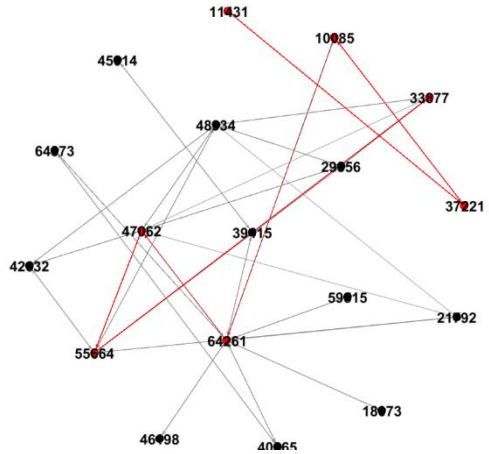


图 3.4 直径为 6 的连通图

参 考 文 献

[1] 汪小帆. 复杂网络理论及其应用[M]. 清华大学出版社, 2006.

[2] Wu, Zhao Lu, Shan-Yuan Ho: Community Detection with Topological Structure and Attributes in Information Networks. ACM TIST 8(2): 33:1-33:17 (2017).

[3] Yang Bo, D Liu , J Liu, D Jin, H Ma. Complex Network Clustering Algorithms. Journal of Software. January 2009, 20 (1).

[4] Barber M J, Clark J W. Detecting network communities by propagating labels under constraints[J]. Physical Review E Statistical Nonlinear & Soft Matter Physics, 2009, 80(2 Pt 2):026129.

[5] Meo P D, Ferrara E, Fiumara G, et al. Fast unfolding of communities in large networks[J]. Journal of Statistical Mechanics Theory & Experiment, 2008, 2008(10):155-168.

[6] GIRVAN M, NEWMAN M E J. Community structure in social and Biological networks [J]. Proceedings of National Academy of Science, 2002, 9(12): 7281-7826.

[7] Newman M E, Girvan M. Finding and evaluating community structure in networks[J]. Physical Review E Statistical Nonlinear & Soft Matter Physics, 2004, 69(2 Pt 2):026113.

[8] Raghavan U N, Albert R, Kumara S. Near linear time algorithm to detect community structures in large-scale networks.[J]. Physical Review E Statistical Nonlinear & Soft Matter Physics, 2007, 76(3 Pt 2):036106.

[9] 邓小龙, 王柏, 吴斌, 等. 基于信息熵的复杂网络社团划分建模和验证[J]. 计算机研究与发展, 2012, 49(4):725-734.

[10] Palla G, Derényi I, Farkas I, et al. Uncovering the overlapping community structure of complex networks in nature and society.[J]. Nature, 2005, 435(7043):814-8