复杂网络的社区挖掘

数学与统计学院　　数学与应用数学专业　　黄金城　　指导老师：刘建熙

摘　要：大量复杂的非结构数据涌现，使得传统的社区挖掘算法在高效处理大型复杂网络上面临困难。针对传统算法存在的收敛速度慢、结果不稳定、随机性强等不足，本文借鉴 Louvain 算法、Tarjan 算法和二点连通等思路，提出二点连通-Louvain 算法，该算法不仅可以进行重叠社区和层次社区的挖掘，还可以基于割点的思想进行重要节点的挖掘。实验结果表明，该社区挖掘算法不仅在社区发现上实现与 Louvain 算法一样的功能，但同时增强了算法稳定性、划分正确率。

关键词：复杂网络；社区挖掘；Louvain 算法；Tarjan 算法；二点连通

Abstract: The emergence of a large number of complex unstructured data makes the traditional community detection algorithms hard to deal with large-scale complex networks efficiently. In view of the shortcomings of traditional algorithms, such as slow convergence speed, unstable results and strong randomness, this paper borrows idea from the Louvain algorithm, the Tarjan algorithm and two-connectivity, and proposes a new community detection algorithm, namely, two-connectivity Louvain algorithm. This algorithm not only detects overlapping communities and hierarchical communities, but also important nodes based on the idea of cut points. The experimental results show that our community detection algorithm not only can have same functionality as that of the Louvain algorithm, but also enhance the stability and accuracy.

Keywords: Complex Network; Community Detection; Louvain Algorithm; Tarjan Algorithm; Two-connectivity.

1 引言

复杂网络(Fortunato, 2007; Porter, 2009; Schaeffer, 2007)是众多个体之间相互作用后产生的系统，是对社会网络(Scott, 2012; Wolfe , 1995)、信息网络(Dourisboure, 2007; Flake, 2002)、技术网络(Krishnamurthy, 2000)、生物网络(Chen, 2006; Rives, 2003)等现实生活中的网络的图形化描述。社区的定义是：复杂网络内部连接紧密且外部连接相对稀疏的一组节点集合，通过社区挖掘、社区划分可以发现复杂网络中由相似属性节点组成的密集集群(李建锋, 2021)。在非重叠社区中，人们将目光局限于节点的单一身份，而近年来随着对现实网络认知的改变，人们察觉单个节点通常会在多个社区中拥有身份，即重叠社区（overlapping community）技术更加符合现实复杂网络的研究需求，因此学术界和工业界逐渐将研究重点移向相关理论研究及其高通量科技实践应用中，并结合生物学、物理学、数学和计算机科学等不同学科的知识背景，分析社区结构可以深刻揭示复杂网络的拓扑结构及功能，大大推进了复杂重叠社区研究领域的技术发展前沿。

当前，学界研究较为深入的社区划分算法大致可分为四类。一类是基于标签传播的重叠社区检测算法，该类算法原理可简述为“物以类聚”——通过吸收节点周边的邻居标签来作为自身的节点标签。以 Raghavan（2007）提出的快速检测网络社区 LPA（Label Propagation Algorithm）算法为例，首先对复杂网络中的每个节点初始化赋予唯一标签，然后通过随机产生的节点序列对节点进行遍历，每个节点选择邻居节点中最多的标签作为自己的标签，随后不断迭代直到收敛来检测社区结构。LPA 的优点在于其线性的时间复杂度以及简单的迭代过程，可以用于快速实现大型复杂重叠网络的社区发现；但同时其划分过程中存在随机性，划分结果存在明显的不稳定性，这也是该类算法的最大弊病。二是基于凝聚式与分割的算法，

该类算法原理可概括为“不断地结合及移除”。以Girvan和Newman提出的经典GN(Girvan Newman Algorithm)算法为例，通过不断移除网络中最大边介数，准确获得有层级结构的社区(hierarchical community)。但该类算法的缺陷在于时间复杂度较高,停止迭代的标准尚并不明确。三是基于模块度优化的算法。以Louvain算法为例，该算法将网络中每个节点视为独立社区，后考虑将新旧模块度正差值最大的每个社区的邻居节点并入该社区，迭代至结果不再改变为止。其明显的缺陷在于，访问节点的不同顺序定将引致不同的结果，同时会在一定程度上影响计算时间，而且还普遍存在模块度分辨率局限(Sato, 2019)，即是在计算过程中会将较小的社区合并从而产生计算误差。另有基于进化算法类的算法，目前研究尚不透彻。

随着大量各类信息数据的出现，传统典型算法已难以满足学界复杂网络的研究需求。加之，尽管目前的重叠社区发现算法已经在探索社区网络中的重叠结构已经达到优良的精度，但这些算法仍然存在相当的优化空间。

本文在充分研究过往算法优劣并进行了大量的调试工作基础上，对过往算法中常见的收敛过慢、结果不稳定、随机性较强等问题，提出了基于模块度优化的多层次 Louvain 算法，旨在充分发挥 Louvain 算法优势的同时，减少其运算时间，并增强其稳定性、准确率。该算法首先使用深度优先遍历的原理，对社区进行二点连通的划分，对二点连通的划分结果中最大社区进行 Louvain 算法划分，并将划分的社区在原社区中聚合成超点，以此作为新社区基本单位以奠定稳定划分的框架。在新的社区框架中，再次进行 Louvain 算法划分，并将前面的超点集还原，获得最终的社区划分结果。由于任何网络中的二点连通子图是稳定且唯一的，加上超点集作为最小社区单元，这便优化了 Louvain 算法对于较小社区合并的缺陷，从而提高了社区划分的稳定性和精确度；并且二点连通划分是基于深度优先的 Tarjan 算法，该算法具有极低的时间复杂度。故在整体上，该算法的时间复杂度也有效降低，从而大大提高了社区划分的效率。

本文算法利用 Zachary 空手道俱乐部网络数据集、线虫代谢网络数据集等大型社区数据，通过构建超点集，再结合 Louvain 算法的思想逐步对社区进行分解，得到具有良好性质的社区结构划分；最后对算法执行时间、正确解释的顶点对数量和模块度等三大关键指标，与其它社区划分的经典算法进行对比。实验表明，本文算法相比作为参照算法的各类算法，都表现出了优秀的计算速度以及准确度。

2 基本概念

在图论中，图是用来表示对象之间的网状关系的重要概念。一个图可以被定义为二元数组：G=(V,E)。其中 V 是顶点集，E 是边集。若在 E 中所有边都是无向的，则称 G 为无向图。本文主要研究无向图的社区划分与发现。

连通图是图论中基于连通性的概念。在一个无向图当中，若顶点 a 与 b 之间存在路径，则可以说顶点 a 与顶点 b 是连通的。若无向图 G 中任意的两个顶点都是连通的，则称该图为连通图。在非连通图中，其中的每个极大连通子图称为无向图的连通分支。一个非连通图中可存在若干个连通分支。

在一个图 G 中，若删除某个顶点（及其所有关联的边）后该图的连通分支数增多，则该点称为图 G 的**割点**。

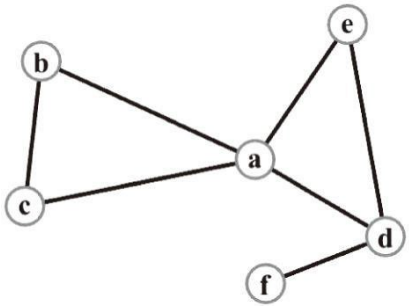

图 1　节点 a 和 d 是该图的割点

在无向图中任意两个顶点间都存在两条节点不相交的路径，或是说该图中删掉任意一个点后还保持连通，则称该无向图是**二点连通图**。换言之，没有割点的无向图就是二点连通图。二点连通图的定义等价于任意两条边都在同一个简单圈内。对于一个无向图，二点连通图的极大子图称为**二点连通分支**。规定两个节点连一条边的图是二点连通图。这样，对于任意一个无向图，其二点连通分支是唯一的。 例如图 1 中存在三个二点连通分支，分别是由节点集{a,b,c},{a,d,e},{f,d}所关联的子图。

3 算法

3.1 概述

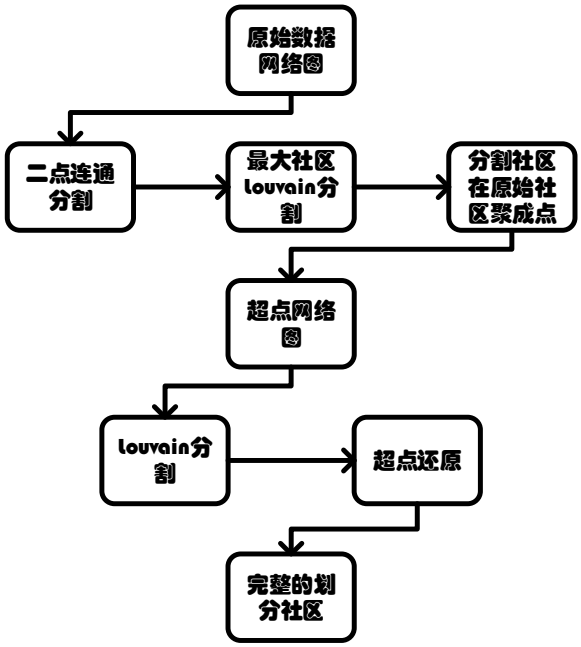

图 2　算法流程图

本文在前人研究的基础上，提出了一个改良性质的 Louvain 算法，该算法目的主要是弱化 Louvain 在小社区合并的缺陷， 同时提升算法效率。但基于技术限制，该算法仅对无向图的数据网络进行社区划分。

构建超点网络，使得原始数据可以按几个数量级的程度进行收缩，同时提升了小社区的在原始社区的权重比例，社区内部耦合度、紧密程度都会得到进一步的提升。

3.2 指标

3.2.1 性能指标

性能P(Fortunato, 2009)的定义：正确解释的“顶点对”的数量，占所有顶点对的比例。即社区划分中，两个顶点属于同一个群落并通过边连接，或两个顶点属于不同群落但是没有边连接的数量占所有可能的边的数量的比例。

即对于图$G=(V,E)$，社区集合$H=\{C_1,C_2,C_3,\dots\}$,对于$\forall i,j\in V$,有：

$$P=\frac{\sum_{i,j}|\{(i,j)\in E,C_i=C_j\}|+\sum_{i,j}|\{(i,j)\notin E,C_i\neq C_j\}|}{n(n-1)/2} \tag{1}$$

论文指出：P取值范围为[0，1]，P 值越大，说明正确“解释”的顶点对的数量越多，算法输出结果越好；$P=1$时当且仅当图是由若干个独立的完全子图构成的。

3.2.2 模块度

模块度Q(Newman, 2006)的定义：落在同一社区划分结果内的边的比例，减去对这些边进行随机分配所得到的概率期望。

即对于图$G=(V,E)$，其有n个节点，m条边。k_v等于节点v的度。对于网络的邻接矩阵A，有：

$$A_{vw}=\begin{cases}0，\ 节点v和w之间没边\\1，\ 节点v和w之间有边\end{cases} \tag{2}$$

对于网络的社区所属矩阵δ，有：

$$\delta_{vw}=\begin{cases}0，\ 节点v和w所属同一社区\\1，\ 节点v和w所属不同社区\end{cases} \tag{3}$$

节点v和节点w存在连接边的期望E_{vw}：

$$E_{vw}=\frac{k_vk_w}{2m} \tag{4}$$

由此，社区划分的模块度Q为：

$$Q=\frac{1}{2m}\sum_{vw}\left(A_{vw}-\frac{k_vk_w}{2m}\right)\delta_{vw} \tag{5}$$

论文指出：一般具有较好社区结构的网络，其 Q 值大致 0.3~0.7 之间，Q 值越高，社区结构越好。

3.2.3 模块度增益

模块度增益ΔQ(Blondel, 2008)的定义：模块度增益设立目的是针对单个节点或社区，当节点 A 或社区 A 合并到社区 B 中时，通过计算新形成的全图的模块度，以及合并前的原始的全图模块度，对比判断此次改动获得的在模块度方面的增益值，由此确定是否保留此次改动。

对于图$G=(V,E)$，其有n个节点，m条边。对于社区i和社区j：$k_{i,in}$表示为社区i与其要移入的社区j内的所有节点之间的边的权重和；k_i表示为社区i和图G上所有的社区的连接的边的权重和；$\sum_{tot}$表示为社区j和图G上所有的社区的连接的边的权重和。

由此，模块度增益ΔQ为：

$$\Delta Q=\frac{1}{2m}\left(k_{i,in}-\frac{\sum_{tot}*k_i}{m}\right) \tag{6}$$

论文指出：$\Delta Q>0$表示模块度增益为正，此次社区改动有利。

3.2.4 调整兰德系数

调整兰德系数ARI(Yeung, 2001)的定义：正确社区划分点对数占所有点对数的比例，通过衡量每个点对在不同的社区划分中，结果是否保持一致来比较社区划分结果与真实划分结果的相似性。

对于真实社区划分$G1$、算法社区划分$G2$：a_{11}表示为$G1$与$G2$内，都属于同一社区的点对数；a_{00}表示为$G1$与$G2$内，都不属于同一社区的点对数；a_{10}表示为$G1$内属于同一社区而在$G2$内属于不同社区的点对数；

a_{01}表示为$G1$内属于不同社区而在$G2$内属于同一社区的点对数。

由此，调整兰德系数 ARI[18]为：

$$ARI = \frac{a_{11} - \frac{(a_{11}+a_{01})(a_{11}+a_{10})}{a_{11}+a_{11}+a_{01}+a_{00}}}{\frac{(a_{11}+a_{01})(a_{11}+a_{10})}{2} - \frac{(a_{11}+a_{01})(a_{11}+a_{10})}{a_{11}+a_{11}+a_{01}+a_{00}}} \quad (7)$$

注意到ARI越大，算法划分和真实社区划分结果更为接近；当$ARI = 1$时，算法划分与真实社区划分结果完全一致。

3.3 深度优先搜索 Depth First Search（DFS）

深度优先搜索算法（DFS）一种用于遍历或搜索树或图的算法。 沿着树的深度对树的节点进行遍历，尽可能深的搜索树的分支。当含有节点 v 的关联边全部被探寻过，或者在搜寻过程中，该结点不满足特定条件，搜索过程将回溯到发现节点 v 的那条边的起始节点。整个过程重复进行，直到所有节点都被访问为止。

3.3.1 算法流程

深度优先遍历使用的是先序遍历的方法，首先遍历当前节点，然后遵循先左后右的顺序进行遍历，同时使用栈来记录将要遍历的节点。遍历当前节点后，将该节点出栈，同时检查该节点是否还有未遍历的邻接节点，如有则从该节点的最右邻接节点开始，将节点入栈，以保证后续是从左邻接点到右邻接点的出栈顺序，如没有则进行回溯，即继续下一个节点的出栈。

以图 3 为例，当前节点为 1，1 先进栈，然后 1 再出栈，完成出栈检查遍历当前节点的邻接节点，存在 2、4，则最右邻接节点即 4 先进栈，然后 2 进栈，至此 1 的邻接节点全部进栈，则继续开始出栈，2 为栈顶先出栈，完成出栈检查遍历 2 的邻接节点，存在 5、6，则依然按顺序 6 先进栈，再 5 进栈。至此 2 的邻接节点全部进栈，此时 5 为栈顶，按照前面规律，最终出栈顺序结果为 1→2→5→6→4，深度优先遍历结束。

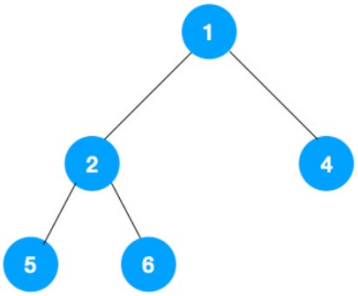

图 3　DFS 算法流程

3.4 Tarjan 二点连通算法

二点连通沿用了 Tarjan 思想，对社区的每个连通分支进行深度优先搜索。

算法维护变量：1、DFS 深度优先栈；2、DFS 顺序访问时间戳 time；3、DFS 深度优先中每个节点的首次访问顺序；4、DFS 深度优先中每个节点在接下来深度优先搜索中可以访问到的最小 dfn，定义为 low。

回溯条件：DFS 深度优先途经节点 v 时，节点 v 的子节点均被访问过，节点 v 的 low 值比其所有子节点的 dfn 值小。

每次 DFS 到一个结点时，将该结点入栈，并记录该节点的 dfn=low=time。遇到回溯条件开始回溯，回溯过程均对每个节点的 low 值更新，更新为包括自己在内的其所有子节点的最小 low 值。回溯过程中若回溯节点的 low 值等于其 dfn 值，就确认了 S 出栈的结束节点为该节点（该结点也出栈，但要吞回，因为割点可重叠），并停止回溯，将此次所有出栈节点计入结果。重复上述操作，直到所有节点被访问过且从 S 出栈。

3.4.1 算法流程

以下图为例，用 Tarjan 算法寻找二点连通分支，设无向图 G=(V,E)如图 4 所示：

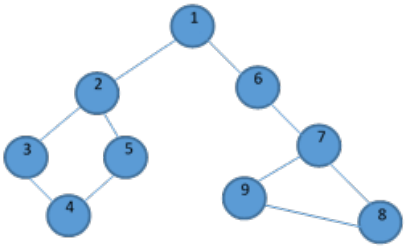

图 4　初始无向图

从节点 1 开始搜索，时间戳 time 也记为 1，节点 1 的 dfn 值和 low 值同为 time 即 1。节点 1 入栈 S，然后默认选择入栈节点最左边未访问子节点，即节点 2 进行深入搜索，此时存在一个关系：节点 2 是节点 1 的子节点，节点 1 是节点 2 的父节点。然后节点 2 入栈 S，同理接下来依次搜索节点 3、4、5，并按顺序将每个节点入栈 S，并对每个节点的 dfn 和 low 赋值。此时 S={1,2,3,4,5}。当搜索来到节点 5，且节点 5 已经进栈，由于节点 5 的子节点均被访问过，此时即为开始回溯的时间点，且节点 5 的 low 值更新为节点 2 的 dfn 值，即 low[5]=2。由节点 5 开始回溯，直到遇到某个节点的子节点存在未访问情况，或者其 low 值等于 dfn 值停止。前者会就这该子节点继续深入搜索，后者将标记当前节点为停止节点，对栈 S 进行出栈操作，直到遇到停止节点（停止节点也要出栈，但回重新进栈）,,后者的优先级大于前者。回溯过程每个当前节点的 low 值更新为与其父节点的 low 值对比的较小值。故节点 4、3 的 low 值均为 2，而由于有 low[2]=dfn[2]，可知节点 2 为停止节点，遂得二点连通分支{2，3，4，5}，S={1，2}；此时回溯到节点 1，而 low[1]=dfn[1]，故又知节点 1 也为停止节点，遂又得二点连通分支{1,2},S={1}。

但由于节点 1 存在另一个未被访问的子节点，故节点 1 向节点 6 搜索，并接着依次搜索节点 7、9、8，此时 S={1,6,7,9,8}，由节点 8 开始回溯，由边（8,7）可更新节点 8 的 low 值为 7，接着回溯，更新节点 9 的 low 值也为 7，而由于 low[7]=dfn[7]，知节点 7 为停止节点，最后找到二点连通分支{7，9，8}，S={1,6,7}；回溯到节点 6，由 low[6]=dfn[6]，又知节点 6 为停止节点，找到二点连通分支{6,7}，S={1,6}；回溯到节点 1，low[1]=dfn[1]，找到二点连通分支{1,6}，S={1}，但节点均被访问，搜索结束。

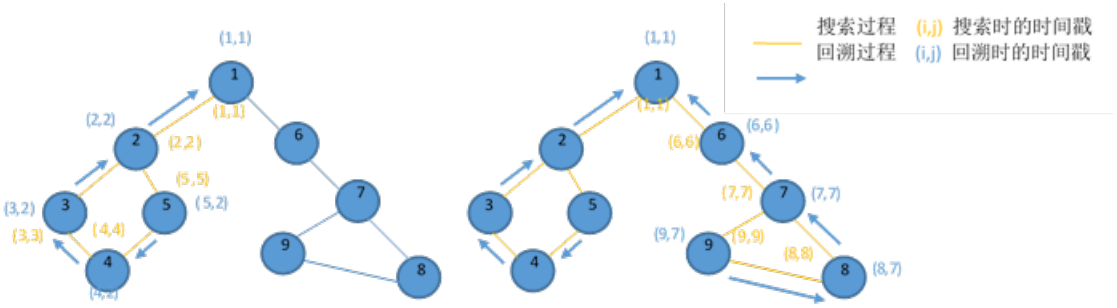

图 5　二点连通寻找过程

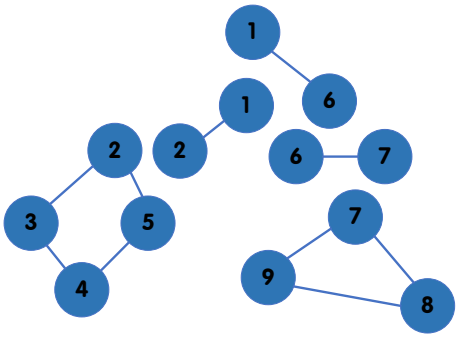

图 6　最终二点连通分支

3.5 Louvain 算法

Louvain 算法是基于模块度最大化的图算法模型。与其他基于模块度和模块度增益算法不同的是，该算法在运行效率上具有显著优势，且对一些节点较多、边较少的图，聚类效果特别明显，能够发现层次性的社区结构。

Louvain 算法主要有三个步骤：1、初始阶段将每个顶点当作一个社区，社区个数与顶点个数相同，这是由下往上的聚合算法的共性；2、依次将每个节点与与之相邻节点合并，计算合并后的模块度增益并统计，最后选择增益值大于零且最大的合并方式合并两个社区；3、将合并社区视作超点，并重复步骤 2，直至不存在增益值大于零的合并方式。

3.5.1 算法流程

以下图为例，使用 Louvain 算法进行社区划分，设无向图 G=(V,E)如图 7 所示：

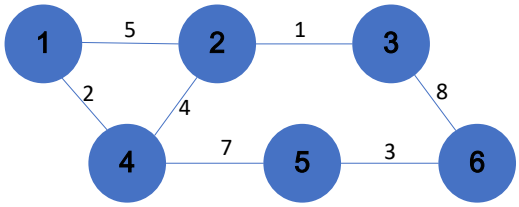

图 7　初始化无向图

令每个节点均为社区，即第一次迭代社区划分结果为：{[1], [2], [3], [4], [5], [6]}；计算每个相邻社区合并的模块度增益，以社区 1 与社区 2 合并为例：

$$\Delta Q_{1,2} = \frac{1}{2*30}\left(5 - \frac{10*7}{30}\right) = 0.0444$$

同理可以计算其他社区合并的值：$\Delta Q_{1,4} = -0.0172$、$\Delta Q_{2,3} = -0.0333$、$\Delta Q_{2,4} = -0.0055$、$\Delta Q_{3,6} = 0.0783$、$\Delta Q_{4,5} = 0.0444$、$\Delta Q_{5,6} = -0.0111$，由此，选择社区 3 和社区 6 作为第一次迭代的合并选择，结果如图 8 所示，第一次迭代完成：

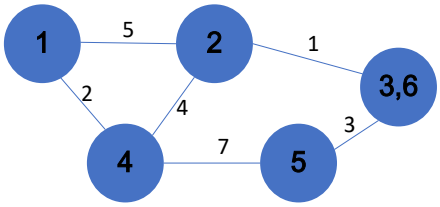

图 8　Louvain 第一次迭代合并

按照上述方式，进行第二次、第三次迭代合并，最终划分过程、结果如图 9 所示：

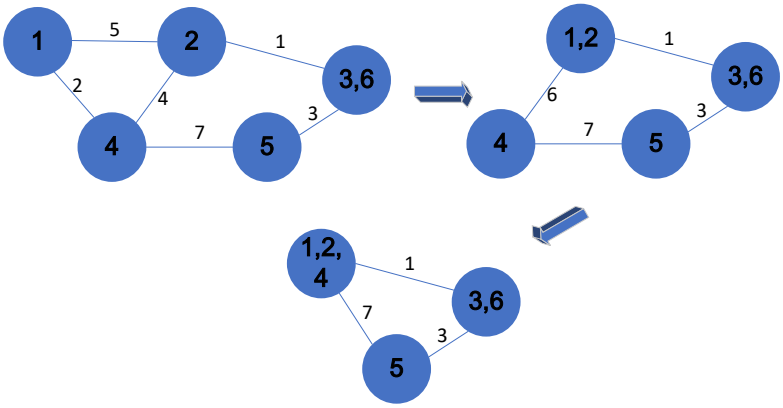

图 9　Louvain 第二次、第三次迭代合并

3.6 二点连通-Louvain 算法

3.6.1 算法思路

Louvain 算法在基于模块度最大化的各类算法中总能在速度和有效性上总体最优，但其算法结果稳定性，小社区错误合并造成过多误差等问题一直被诟病。在 Louvain 算法发布之后，有无数前人都试图解决 Louvain 算法的这类问题，并有所进展，如：过滤权重较弱的边、修改模块度增益公式增加阈值、可视化迭代人为确定迭代停止时机等等。但这类优化大部分基于 Louvain 算法本身过程进行的，有一定的局限性。

本文的思路是首先对划分社区数据进行一定的优化，通过获取二点连通算法的划分社区，令其在原社区完成第一次聚点，使得划分结果不会再发散分离，此目的是利用二点连通算法在社区划分中的绝对稳定性和结构一致性，首先确立第一批稳定社区。此操作在增强算法整体稳定性的同时，有效减少后面 Louvain 算法对小社区错误合并的可能性，因为小社区大部分已经在二点连通已经被正确聚合。故在此基础上，再次进行 Louvain 算法对社区进行二次划分，就能获取更好的结果。

鉴于上述理解，代码设计得以完成，但是实验结果却没有如构思所料。通过对算法每一步的研究分析。我们研究发现，欲划分的社区大部分均为较为紧密的社区，对社区进行二点连通划分后，会出现一个较为突出的超大规模子社区，将该社区聚成超点显然会使得后面的划分崩溃。因此，在对原图进行二点连通分支划分的基础上，对该超大规模二连通分支的子社区，首先进行 Louvain 算法分割，并将该分割融入到二点连通的分割中，这并没有违反本文的初衷，因为超大规模子社区本身已经剔除了大部分的小社区，故对其进行 Louvain 算法分解时，小社区错误合并带来的误差会大大减小。

同时，基于对社区的理解，了解到社区的节点总有主次之分，每个节点的重要程度并不相同。显然，割点的重要程度相对于其他节点来说更为重要，因为它是连接两个社区的枢纽，无论是在通讯网络，还是铁道网络，割点重要性毋容置疑，一旦破坏，将会导致两个社区交流的瘫痪。而二点连通算法，可以同时获取社区的割点，这弥补了 Louvain 算法在核心节点挖掘的缺陷。

3.6.2 算法流程

以图 10 为例，使用空手道数据集进行演示。使用二点连通-Louvain 算法进行社区划分：

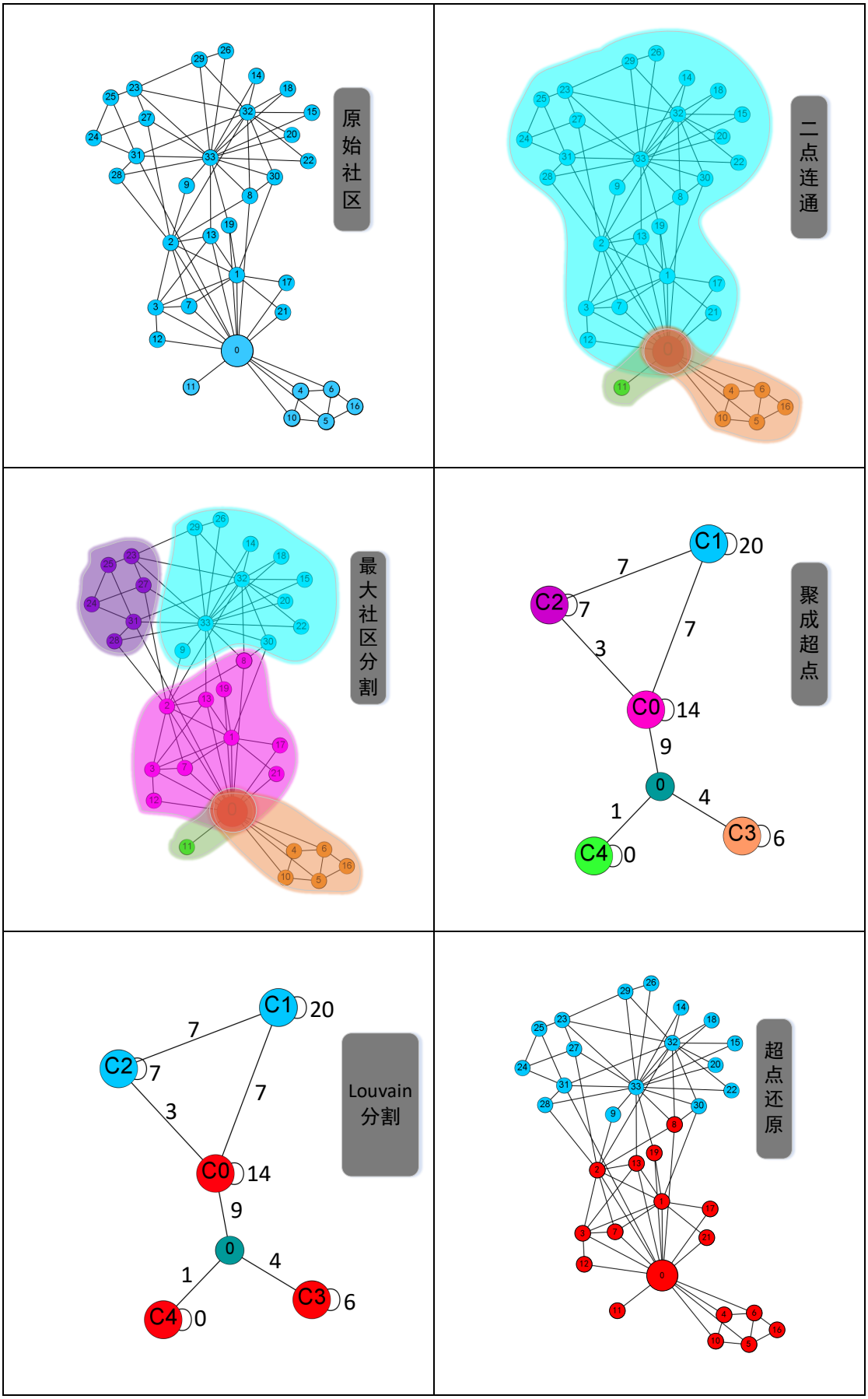

图 10　二点连通-Louvain 算法演示

通过二点连通，可以将社区分割成蓝色社区、绿色社区、橙色社区三个子社区，其中节点 0 是它们的割点，它起到三个子社区的连接作用，在真实的空手道网络中，它是这三个社区的中间人。

通过最大社区分割，聚成超点，可以看出：社区的数量从 34 降到了 5。这种降低在空手道类型的小社区并不明显，但也起到了作用，而在大社区中，这种降低将是多个数量级的降低，会更为显著。

最后 Louvain 分割，超点还原，完成社区划分。计算得出二点连通-Louvain 算法的$ARI_{my}=1$，对比 Louvain 算法的$ARI_{Louvain}=0.50807$，提升效果显著。并且在图 11 中，将本文的算法社区划分和真实社区划分进行对比，可以发现：本文的算法社区划分结果和真实社区划分结果是完全相同的，取得了空手道数据集算法划分结果的最好结果，这也进一步表明我们算法的优越性。

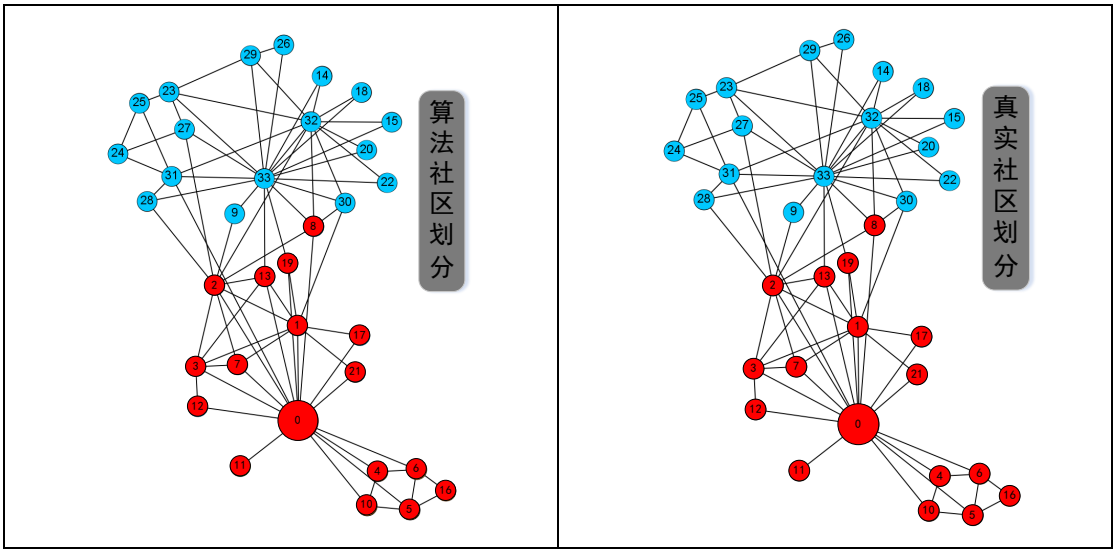

图 11　算法社区划分和真实社区划分对比

4 实验

4.1 实验环境

系统	Window10 64 位 专业版
处理器(CPU)	i7-8750H
内存(RAM)	16GB
语言	Python
软件	PyCharm 2020 社区版

表 1　本文实验环境

4.2 实验数据

本文将在多个数据集中进行一系列实验，其中有些数据集包含真实的社区划分数据，如 Zachary 空手道俱乐部网络数据集(Zachary, 1977)、线虫代谢网络数据集(Duch, 2005)、亚马逊产品联购网络和真实社区数据集(Yang, 2015)；有些数据集为社交网络数据集，如 Facebook 社交网络数据集(McAuley, 2012)、Deezer 克罗地亚用户友谊网络数据集(Rozemberczki, 2019)、Pokec 在线社交网络数据集(Rossi, 2015)。

数据集	数据集信息	节点数	边数

Zachary 空手道俱乐部网络	复杂网络分析中最经典的网络，每个节点代表空手道俱乐部的成员，节点之间存在边表明他们是朋友关系。后来由于俱乐部中的教练和主管发生争执使得整个网络分成两个社团。	34	78
线虫代谢网络	线虫秀丽隐杆线虫的代谢网络。节点是代谢物（例如蛋白质），边是代谢物之间的相互作用。由于代谢物可以与自身相互作用，因此网络包含循环。交互是无方向的，任何两种代谢物之间都可能存在多种相互作用。本文删去自环边。	453	4574
Facebook 社交网络	该数据集由来自著名社交软件 Facebook 中的“圈子”（或“朋友列表”）组成。该 Facebook 数据是从使用 Facebook 社交软件的调查参与者中收集。	4039	88234
Deezer 友谊网络——克罗地亚国家	数据来自音乐流媒体服务 Deezer（2017 年 11 月）。这些数据集代表了来自 3 个欧洲国家的用户的友谊网络。节点代表用户，边代表相互的友谊。在每个数据集中，用户可能喜欢 84 种不同的流派。喜欢的流派列表是根据喜欢的歌曲列表编制的。本文选用其中的克罗地亚国家的 Deezer 友谊网络数据集。	54,573	498,202
亚马逊产品联购网络和真实社区	网络是通过抓取亚马逊网站收集的。它基于亚马逊网站的购买此商品的客户也购买了功能。如果产品 i 经常与产品 j 共同购买，则图中包含从 i 到 j 的无向边。亚马逊提供的每个产品类别定义了每个真实社区。	334863	925872
Pokec 在线社交网络	Pokec 是斯洛伐克流行的在线社交网络。该数据集包含用户关系列表。	1632803	22301964

表 2　本文所使用数据集的信息

4.3 实验结果

在实验中采取最为关键的三个评价指标进行算法对比，一是执行时间，适用于所有的数据集，二是性能指标 P——正确解释的“顶点对”的数量，该值越大，社区划分越明朗，社区越稠密，社区划分结果越好，三是模块度 Q——该值越大，社区划分结果越好。

本文选取社区划分中比较经典的算法与本文的二点连通-Louvain 算法(简称 TL 算法)进行对比,如下：基于模块度的 Clauset-Newman-Moore(CNM)算法(Clauset, 2004)、Louvian(LV)算法(Xie, 2011)；基于中心性概念的 Girvan－Newman(GN)算法(Girvan, 2022)；基于标签传播的 Label Propagation Algorithm(LPA)算法(Raghavan, 2007)、Speaker-Listener LPA(SLPA)算法(蔡国永, 2013)、半同步 LPA(ISLPA)算法

(Cordasco, 2010)；异步流体的 Fluid Communities(FC)算法(Parés, 2017)（社团数(K)采用特定函数获取：$K = \lfloor (1.7^{\ln(N)}) \rfloor$, N为数据集节点数，$\lfloor\ \rfloor$为向下取整符号），性能、运行时间对比如下表所示：

数据集	TL	CNM	LV	GN	LPA	SLPA	ISLPA	FC
Karate	0.61676	0.71480	0.80392	0.61141	0.78253	0.59537	0.38681	0.85383
线虫	0.84997	0.77670	0.86355	0.07135	0.22260	0.75315	0.15886	0.95180
Facebook	0.91407	0.85873	0.91553	------	0.95162	0.85160	0.86793	0.98400
Deezer	0.93627	-------	0.93579	------	0.92091	------	0.94609	0.99676
亚马逊	0.98558	-------	0.98574	------	0.99994	------	------	0.99872
Pokec	0.91737	-------	0.93075	------	0.94603	----	----	0.99946

表 3　算法性能表

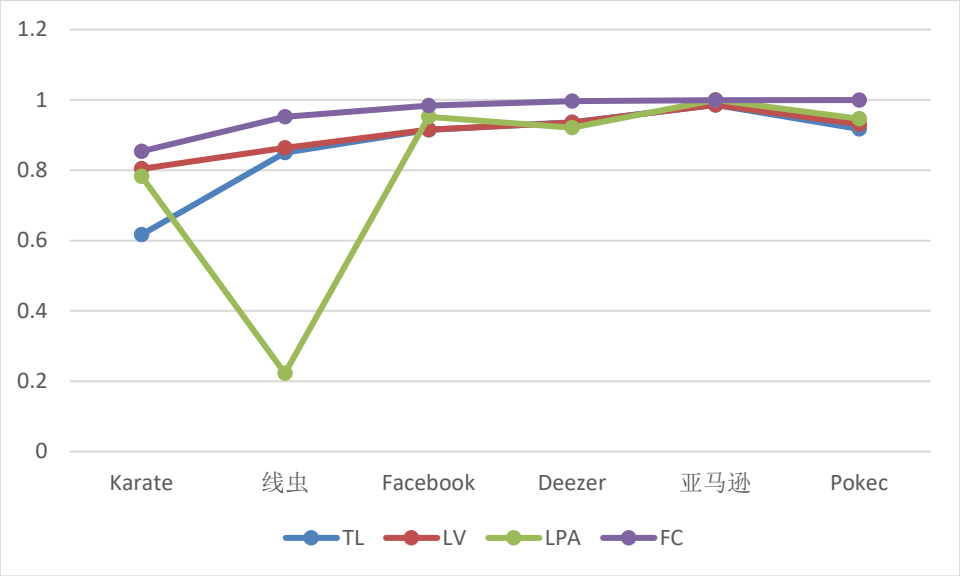

图 12　算法性能图示

表 3 与图 12 展示了各个算法在真实数据集上的性能指标 P 的比较结果。由结果可知，TL 算法在 6 个数据集中，基本都获得了比较好的结果。其中，在处理数据量较小的网络时，LPA 算法的性能较弱，且不稳定，而 FC 算法的结果在数据集开始庞大的情况下，性能指标的结果越发与 TL 算法的结果更为接近，甚至有略微优过的迹象，总体良好。但是 FC 算法需要预先指定社团数，不同的社团数会导致不同的结果，导致 FC 算法结果的不稳定性更加强烈，相比之下，TL 算法是在第一层设定了二点连通的框架，该框架将孤立点以及连通分支分开，保证原社区两点之间的连接，以最少的断边方式，获得结构更为紧密的社区结构，避免分支内部关系缺乏，丧失应用价值，同时其输出结果是极其稳定的。这强大的稳定性确保了后续 Louvain 算法建立在牢固社区划分基础之上。

数据集	TL	CNM	LV	GN	LPA	SLPA	ISLPA	FC
Karate	0.01600	0.01200	0.00504	0.09298	0.00304	0.06304	0.00296	0.00395
线虫	0.22201	1.21508	0.19527	18.46991	0.06789	1.71015	0.03971	0.08304
Facebook	5.63009	73.48866	4.69094	------	2.94367	64.59069	0.89463	6.91093
Deezer	93.25196	-------	106.0551	------	80.16602	------	29.85076	78.92753
亚马逊	225.9062	-------	222.5620	------	135.4165	------	------	214.4485
Pokec	7420.16	-------	6943.54	------	3838.29	------	------	2616.09

表 4　算法执行时间表

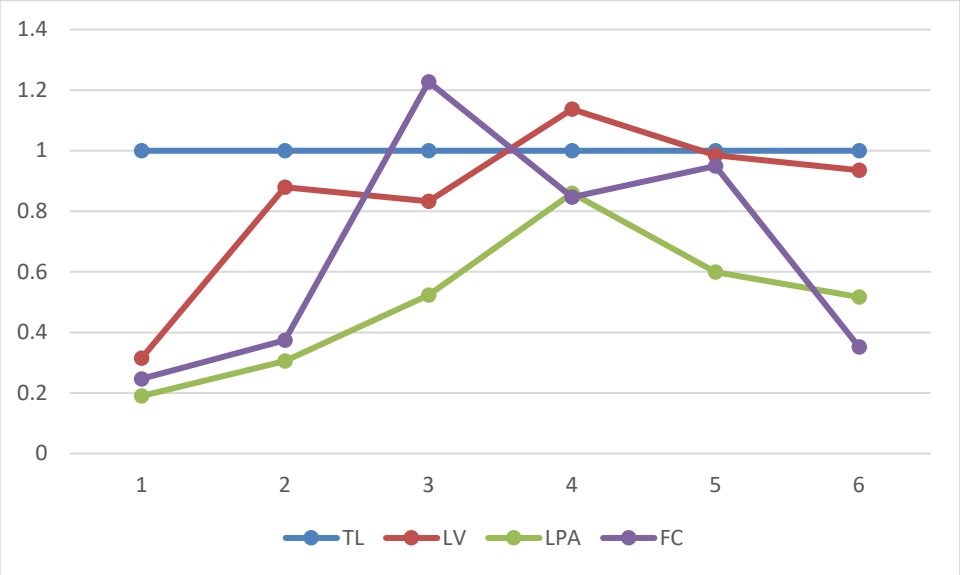

图 13　算法执行时间图示(相对 TL 百分比)

表 4 与图 13 展示了各算法在真实网络数据集中的平均运行时间，时间单位为秒（s）。在该表中，TL 算法在所有数据集的处理中均占优势，尤其在面向大型社交网络的情况下，社交网络越大，TL 算法的优势就越发明显， 但是，由于 TL 与 Louvain 算法捆绑，即使基于 Tarjan 思想的二点连通算法具有非常低的时间复杂度，也没能很好抵消 Louvain 算法的执行时间依赖。在结构紧密的社区中，二点连通算法并不能很好的将适量的二点连通分支聚成超点，从而导致超大规模子社区涉及的节点数量过多，Louvain 算法的运行占比过大。相比之下，对于稀疏的 Deezer 网络，TL 算法对比 Louvain 算法的运行耗时有较为明显的下降趋势，下降比例为 12.07%。

数据集	TL	CNM	LV	GN	LPA	SLPA	ISLPA	FC
Karate	0.37146	0.38067	0.41978	0.35996	0.41511	0.33432	0.11210	0.33670
线虫	0.43647	0.40158	0.43460	0.02814	0.07045	0.24340	0.05093	0.34637
Facebook	0.83490	0.77453	0.83494	------	0.82241	0.69320	0.73684	0.52319
Deezer	0.73795	-------	0.73966	------	0.68568	------	0.67376	0.47580
亚马逊	0.92624	-------	0.92639	------	0.70979	------	------	0.83575
Pokec	0.71576	-------	0.71642	------	0.68328	------	------	0.33374

表 5　算法模块度表

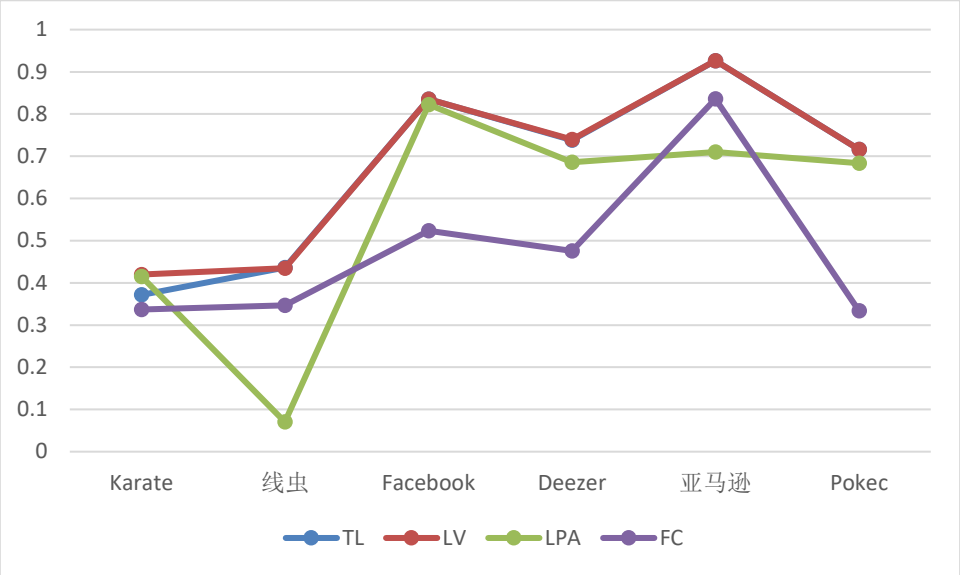

图 14　算法模块度图示

表 5 与图 14 展示了各算法在正式网络数据集中的平均模块度的比较结果。在该表中，可以明显的发现：除了空手道俱乐部网络，TL 算法的模块度一直和 Louvain 算法的模块度保持高度相似，这说明加入二点连通算法层的框架，并没有显著的影响了 Louvain 算法的本身性能；相反，二点连通算法框架在保持了 Louvain 自身优势的前提下，利用自身的特性，增强了算法的稳定性，并且由前面空手道网络的算法描述中，可以了解到，增加了二点连通算法框架，能非常显著的提升算法划分社区与真实划分社区的相似度、贴合度，这正是本文设计该算法的目的——保证 Louvain 算法优势的前提下，提高其稳定性，弱化其对小社区合并的误差。

5 结语

本文基于对传统算法的认识以及对 Tarjan 算法、Louvain 算法的深刻理解，在前人研究的基础上，提出了二点连通-Louvain 的社区挖掘算法，运行效率及精确度更高，稳定性更强。该算法基于多层角度的思考，基于 Tarjan 算法的思想，设计二点连通的算法框架层，再利用 Louvain 算法的优势，最终得到具有良好性质的社区结构划分。为验证该算法的正确性和效果，本文对多个不同类型的数据集进行社区划分实验，结果表明该算法的效果与传统算法相比较为理想，能够在保持 Louvain 算法优势的基础上划分出更具优秀结构的社区。同时，该算法能够弥补 Louvain 算法在小社区划分中准确度过低的局限，以近乎完美的 ARI 得分完成了 Karate 空手道网络的划分。

参考文献:

[1]李建锋,卢迪,李贺香.一种改进的原子搜索算法[J/OL].系统仿真:1-13[2021-04-10].

[2]蔡国永, 林航, 文益民. 社会语义网社区发现标签传递算法研究[J]. 计算机科学, 2013(02):53-57.

[3]Fortunato S., Barthelemy M., Resolution limit in community detection[J]. Proceedings of the National Academy of Sciences, 2007, 104(1):36-41.

[4]Porter M. A., Onnela J. P., Mucha P. J., Communities in Networks[J]. Notices of the AMS, 2009, 56(9):4294–4303.

[5]Schaeffer S. E., Survey: Graph clustering[J]. Computer Science Review, 2007, 1(1):27-64.

[6]Scott.J., Social Network Analysis Sage[M]. SAGE Publication Ltd, 2012.

[7]Wolfe A. W., Social Network Analysis: Methods and Applications, by Stanley Wasserman; Katherine Faust[J]. Contemporary Sociology, 1995, 91(435):219-220.

[8]Dourisboure. Y., Geraci. F., Pellegrini. M., Extraction and classification of dense communities in the web, Proceedings of the 16th international conference on World Wide Web, 2007, 461-470.

[9]Flake G. W., Lawrence S., Giles C. L., et al. Self-organization and identification of Web communities[J]. Computer, 2002, 35(3):66-70.

[10]Krishnamurthy B., Wang J., On network-aware clustering of Web clients[J]. ACM SIGCOMM Computer Communication Review, 2000, 30(4):97-110.

[11]Chen J., Yuan B., Detecting functional modules in the yeast protein-protein interaction network[J]. Bioinformatics, 2006, 22(18):2283-2290.

[12]Rives A. W., Galitski T., Modular organization of cellular networks[J]. Proceedings of the National Academy of Sciences, 2003, 100(3):1128-1133.

[13]Raghavan U. N., Albert R., Kumara S., Near linear time algorithm to detect community structures in large-scale networks[J]. Physical review E, 2007, 76(3):036106.

[14]Girvan M.,Newman M.E.J., Community structure in social and biological networks[J].Proceedings of the National Academy of Science of the United States of America, 2022,99(12):7821-7826

[15]Sato K.,Izunaga Y., An enhanced milp-based branch-and-price approach to modularity density maximization on graphs[J]. Computers and Operations Research, 2019, 106:236-245

[16]Fortunato S., Community Detection in Graphs[J]. Physics Reports, 2009, 486(3-5).

[17]Newman M., Modularity and community structure in networks[J]. Proceedings of the National Academy of Sciences of the United States of America, 2006, 103(23):8577-8582.

[18]Blondel V. D., Guillaume J. L., Lambiotte R, Fast unfolding of communities in large networks[J]. Journal of statistical mechanics: theory and experiment, 2008, 2008(10): P10008.

[19]Yeung K. Y., Ruzzo W. L., Details of the Adjusted Rand index and Clustering algorithms Supplement to the paper "An empirical study on Principal Component Analysis for clustering gene expression data" (to appear in Bioinformatics). 2001.

[20]Zachary W. W., An information flow model for conflict and fission in small groups[J]. Journal of anthropological research, 1977, 33(4): 452-473.

[21]Duch J., Arenas A., Community detection in complex networks using extremal optimization[J]. Physical review E, 2005, 72(2): 027104.

[22]Yang J., Leskovec J., Defining and evaluating network communities based on ground-truth[J]. Knowledge and Information Systems, 2015, 42(1): 181-213.

[23]McAuley J. J., Leskovec J., Learning to discover social circles in ego networks[C]//NIPS. 2012, 2012: 548-56.

[24]Rozemberczki B., Davies R., Sarkar R., Gemsec: Graph embedding with self clustering[C]//Proceedings of the 2019 IEEE/ACM international conference on advances in social

networks analysis and mining. 2019: 65-72.

[25]Rossi R., Ahmed N., The network data repository with interactive graph analytics and visualization[C]//Twenty-Ninth AAAI Conference on Artificial Intelligence. 2015.

[26]Clauset A., Newman M. E. J., Moore C., Finding community structure in very large networks[J]. Physical review E, 2004, 70(6): 066111.

[27]Xie J, Szymanski B. K., Liu X., Slpa: Uncovering overlapping communities in social networks via a speaker-listener interaction dynamic process[C]//2011 ieee 11th international conference on data mining workshops. IEEE, 2011: 344-349.

[28]Cordasco, G., & Gargano, L. (2010, December). Community detection via semi-synchronous label propagation algorithms. In Business Applications of Social Network Analysis (BASNA), 2010 IEEE International Workshop on (pp. 1-8). IEEE.

[29]Parés F., Gasulla D. G., Vilalta A., Fluid communities: A competitive, scalable and diverse community detection algorithm[C]//International Conference on Complex Networks and their Applications. Springer, Cham, 2017: 229-240.

导师评语：

复杂网络是众多个体之间相互作用后产生的系统。社区是复杂网络内部紧密连接而外部连接相对稀疏的一组节点集合，通过社区挖掘可以发现复杂网络中由相似属性节点组成的密集集群. 本文选题社区挖掘在多个学科中具有重要的理论和实践意义。本文写作规范，提出了全新的社区挖掘算法，较经典社区挖掘算法更快速、更稳定，且能完美复现经典的空手道网络的社区结构，在多个网络中社区挖掘效果优良。论文体现了该生扎实的数学基础、应用数学解决实际问题的能力，也体现了该生很强的编程能力。本文符合学校关于本科生毕业论文的要求。