

本 科 生 毕 业 论 文



题目 基于通话记录的动态社会网络

团体发现与分析算法研究

姓 名 周 晓 龙

学 号 3071102314

指导教师 何 钦 铭

专 业 计算机科学与技术

学 院 计算机科学与技术学院

A Dissertation Submitted to Zhejiang University for the Degree of Bachelor of Engineering



TITLE: An Approach to Discover and Analyze Communities in Dynamic Social Networks

Based on Call Records

Author: XIAO LONG ZHOU

Supervisor: QIN MING HE

Major: Computer Science and Technology

College: College of Computer Science

Submitted Date: 2011.06.01

浙江大学本科生毕业论文（设计）诚信承诺书

1.本人郑重地承诺所呈交的毕业论文（设计），是在指导教师的指导下严格按照学校和学院有关规定完成的。

2.本人在毕业论文（设计）中引用他人的观点和参考资料均加以注释和说明。

3. 本人承诺在毕业论文（设计）选题和研究内容过程中没有抄袭他人研究成果和伪造相关数据等行为。

4. 在毕业论文（设计）中对侵犯任何方面知识产权的行为，由本人承担相应的法律责任。

毕业论文（设计）作者签名：

年 月 日

摘要

人与人之间的互相联系使我们的社会形成一张巨大的随时间动态变化的关系网络。按照共同兴趣或目的可以将人们划分成不同的团伙。研究社会网络中团伙的发现技术和团伙的演化规律对于社会学、营销学、刑侦学等领域都有重要的意义。而人与人之间的联系表现在个人之间的通话记录、Email记录、共同活动等方面。本文就将立足于通话记录，研究实现实际应用需要的团伙发现算法和团伙演化临界事件侦测算法。最后，本文还将把文中提到的算法体系在一个通话记录数据中进行实践。

关键词　数据挖掘、社会网络、聚类算法

Abstract

**Keywords**Data Mining, Social Network, Clustering Algorithm

目录

[摘要 I](#_Toc294445816)

[Abstract II](#_Toc294445817)

[目录 III](#_Toc294445818)

[第1章 绪论 5](#_Toc294445819)

[1.1 课题背景 5](#_Toc294445820)

[1.2 相关工作 7](#_Toc294445821)

[1.3 本文主要工作和意义 8](#_Toc294445822)

[1.4 本文组织结构 9](#_Toc294445823)

[第2章 基于通话记录的动态社会网络团体发现与分析问题描述 10](#_Toc294445824)

[2.1 团伙发现 10](#_Toc294445825)

[2.2 团伙演化临界事件侦测 11](#_Toc294445826)

[第3章 团伙发现算法 12](#_Toc294445827)

[3.1 背景概述与问题描述 12](#_Toc294445828)

[3.2 Betweenness算法 12](#_Toc294445829)

[3.3 Modularity度量与自动聚类 16](#_Toc294445830)

[3.4 基于特定团伙的分析——扩散-剪枝算法框架 20](#_Toc294445831)

[3.5 实验设计与分析 22](#_Toc294445832)

[3.6 本章小结 29](#_Toc294445833)

[第4章 团伙演化的临界事件侦测算法 30](#_Toc294445834)

[4.1 背景概述与问题描述 30](#_Toc294445835)

[4.2 临界事件侦测算法 36](#_Toc294445836)

[4.3 实验设计与分析 38](#_Toc294445837)

[4.4 本章小结 44](#_Toc294445838)

[第5章 一个基于通话记录的应用实例 45](#_Toc294445839)

[5.1 背景概述与问题描述 45](#_Toc294445840)

[5.2 基于通话数据的数学建模 45](#_Toc294445841)

[5.3 实验设计与分析 47](#_Toc294445842)

[5.4 本章小结 50](#_Toc294445843)

[第6章 总结与展望 51](#_Toc294445844)

[参考文献 52](#_Toc294445845)

[致谢 54](#_Toc294445846)

绪论

课题背景

社会中人与人之间的错综复杂的关系，可以抽象成一个非常复杂的网络图，每个人就是这个图中的一个节点，而人与人之间的关系就是这个图的边。人和人的关系有陌生有紧密，紧密的关系（如共同的兴趣、共同参与某类事件）使相关人们形成一个团体。研究社会网络中的团体挖掘算法、团伙的进化规律对社会学研究和相关应用有非常重要的意义。例如，对犯罪分子的犯罪网络团体的挖掘和研究能够帮助公安机关深层次地挖掘埋藏在人际关系中的重要线索，辅助公安刑侦人员的案件侦破。

通常来说，随着时间的变化，社会网络中的团体构成随时都在发生着变化。相关的团体可能发生如分裂、合并、生长、消亡等变化。研究动态的社会网络中团体的变化规律，有很好的应用前景，如帮助社会学家研究社会发展的相关规律，帮助商家和受众掌握最新的流行趋势，帮助政府部门根据社会的发展制定最优决策；研究互联网社区(如微薄、论坛等)团体的进化，有助于相关组织者把握最新的流行趋势，更好地服务参与者(如微薄用户、电子商务消费者等)，也能对商业决策提供有力的支持。

将社会中人与人之间的联系表现为人们的电话通信记录、邮件记录、共同参加一个活动的记录等等，使用一定的算法和数学模型对这些数据进行建模，可以把社会网络抽象为一个网络图。如图1-1所示的就是一个Blogsphere博客社区的关系网络图，每个点代表一个博客，每条边代表博客文章的引用和评论关系，这个网络的结构随着博客社区的演化发生演化。

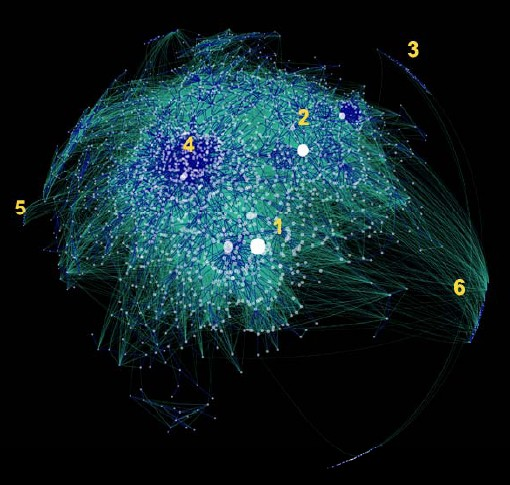


图1- 1 Blogosphere社区的博客引用网络图

研究基于这些网络图上的团体发现和分析的相关算法，并开发有关的计算机程序，可以自动化地高效地分析数据，挖掘出有用的信息和规律。如1-2所示的是一个研究者合作研究的社会网络，节点代表科学家，边代表他们合作参与共同研究的关系。通过一些社会网络挖掘的算法，可以将具有共同兴趣趋向的科学家聚集成团体，图中用不同的颜色代表不同的团体。

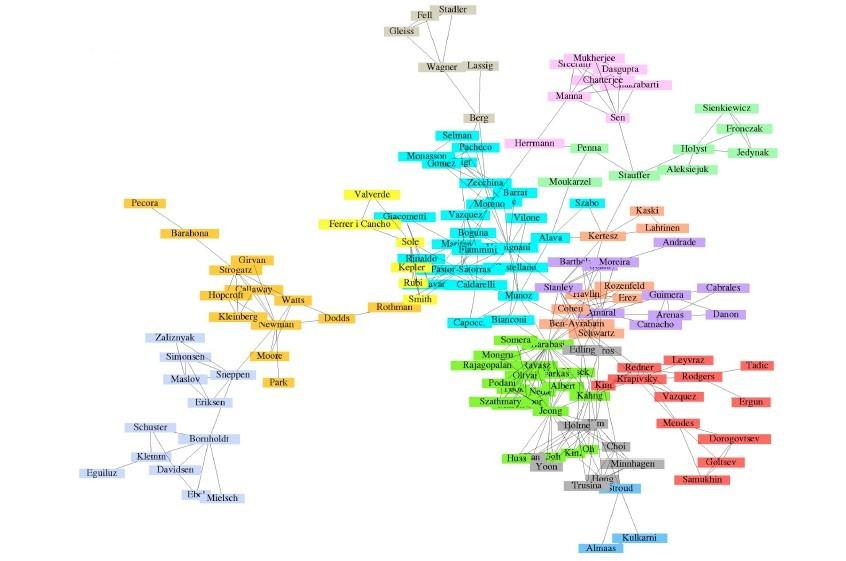


图1- 2研究合作作者关系网络，社会网络的一个示例

我们的研究数据集对象——通话记录中，包含着非常丰富的人际关系信息。关系紧密（参与共同活动、社团关系等）的人之间有比较频繁的通话联系。对通话记录进行建模，以个人为节点，通话记录（包含频率和时间等）归一化为节点之间边的权值，可以将基于通话记录的人际关系网络抽象为一个社会网络图。随着时间的变化，这个网络图的结构也在发生着变化，这是一个动态网络图。利用这个人际关系网络图，我们可以为公安机关的团伙案件侦破提供有力的线索。我们将研究和应用动态网络的挖掘方法，对基于通话记录的人际关系社会网络进行挖掘，发现有用的团伙的信息和演化规律。

相关工作

对于静态的网络进行团体发现和挖掘，已经有相当多年的研究和很多成熟的算法，如层次聚类、k-means聚类、betweenness切片划分算法[9]等等。这些算法能够根据不同的聚类相似度依据对静态网络进行分析并挖掘出其中的团伙。对于静态网络，也有很多的研究和方法来分析团伙内成员、团伙间的联系。关于静态网络的数据挖掘现在已经有非常成熟的知识体系和方法。但是，现有的静态网络的挖掘方法对实际社会网络分析的支持还并不充分，因为实际的社会网络，总是处在时刻变化中。如果有一套行之有效的对于动态网络的分析挖掘方法，将为相关领域的应用提供非常大的帮助。

对于动态网络的挖掘研究，近几年来才刚刚起步，但也已经受到相当大的关注。最初的研究来自Hopcroft等人[11]。他们的研究中定义了“自然团”（随时间推移比较稳定的团）的概念，并使用最佳匹配的方式追踪不同时间片的团来分析团的演化现象。Asur等人的研究中[3]，提出了事件（Event）的概念，定义了团伙的延续、合并、分裂、生成、分解事件，并提出了定量的数学模型。Palla提出[4]可以根据团伙在前面几个时间片上的行为预测在以后时间上的行为趋势。

本文主要工作和意义

发现特定定义多维空间中籍由相似特性“聚集”在一起的团伙（类簇），研究这些团伙演化的历程和规律，早就以其在社会学、生物学、地理学等领域的诱人应用前景引起了人们的关注。对用于发现团伙的聚类算法研究，已经有数十年的历史，也已经有诸如层次聚类、k-means聚类、betweenness切片聚类等成熟的聚类算法。

然而，没有任何一种聚类技术(聚类算法)可以普遍适用于揭示各种多维数据集所呈现出来的多种多样的结构[11]。对于不同的应用场景，我们需要选择合适的技术来对数据集进行分析从而挖掘出有意义的信息。在实际应用中，有相当一部分场景，是需要研究发现一个（或一些）具有特定共性特征的团伙的，而并不关心其他团伙的情况。例如：在某公司的市场调研中，需要研究发现对公司某类产品感兴趣的客户群体，研究这类群体的演化以把握市场的动态和趋势，从而制定出有效的营销策略；在公安机关侦破团伙性案件时，需要研究并发现某个（或某几个）犯罪嫌疑人所在的团伙，分析团伙的演化过程，从而为案件的侦破提供有力的支持。本文针对类似的场景，选取适宜的聚类技术，并提出一种扩散-剪枝的团伙发现算法框架，为实际应用提供理论支持。

在实际的社会网络中，人与人之间的联系时时刻刻都在发生着变化，籍由共同特性或兴趣爱好“聚集”在一起的类簇（团伙）也在时时刻刻发生着演化。对动态网络的研究，虽然不比聚类算法的研究历史悠久，但也已经有很多非常精彩的理论和算法提出。本文重点研究Asur等人提出的基于事件(Event)[3]的团伙行为发现算法。除了Asur提出的团伙Continue延续、k-Merge合并、k-Split分裂、Form生成、Dissolve分解事件，本文还扩增提出k-Continue延续、k-Grow扩张、k-decline衰弱事件，更为丰富地描述团伙演化过程中的临界事件。

本文还将使用文中总结的算法框架就一个通话记录数据集进行分析，发现其中的特定团伙，追踪其演化过程，发现团伙的临界事件。

本文组织结构

本文第2章将总体介绍团伙发现和临界事件侦测的问题描述和基本定义。第3章将重点介绍基于我们的应用场景而选取的Betweenness切边算法，并进一步介绍用于度量聚类效果的Modularity度量算法并用于实现一个自动化聚类的算法框架，最后提出一种用于发现和分析特定团伙的扩散-剪枝算法框架。第4章将描述在团伙动态演化过程中的临界事件的定义以及实现一种发现临界事件的算法并进行实验验证。本文第5章将利用本文提到的算法框架对一个实际的通话记录数据集进行建模分析，发现特定的团伙并追踪其演化过程中的临界事件。第6章对本文的工作进行了总结并提出一些未来的研究方向。

基于通话记录的动态社会网络团体发现与分析问题描述

## 团伙发现

社会中的个人以及人与人之间的联系可以抽象为一张巨大的社会网络图。个人就是这个网络图中的节点(Vertex)，而个人间的联系就是这个网络图中的边(Edge)。具有同样兴趣或者共同从事某项活动等共性使人们之间的联系变得紧密，并形成团伙(cluster)。发现社会网络中的团伙对社会学、营销学、心理学、刑侦学等领域的研究都有重要的意义。

要发现网络中的团伙，就要用到聚类技术（算法）。所谓聚类，就是一个无监督的分类过程，它没有任何先验知识可用，仅根据数据集（目标网络）的结构进行分类活动。其一个形式化的描述如下：

设V={v1, v2, .., vn}表示网络中的点集，设C={C1, C2, ..,Ck}是V的一个子集的集合，,i=1,2,..,k。设dist(vi,vj)表示vi和vj之间的距离。若C是V下的一个聚类结果，那么需要满足以下条件：

1. 对于任何Cp, Cq , 有

条件1保证聚类结果覆盖网络中的所有节点，条件2保证聚类无重叠（当然现在也有很多重叠聚类的研究），条件3保证同一个类簇内的任何两个节点的距离小于任何两个不同类簇间的节点的距离。

关于聚类算法的研究已经有相当长的历史了，也已经有了很多成熟的聚类算法，大致可以分为层次聚类算法、划分式聚类算法、基于密度和网格的聚类算法和其他的聚类算法几类。然而，对于各种各样不同的数据集和不同的应用场景，不可能有哪种聚类的算法能够普遍地适用于所有的情形。在我们的应用场景中，所面对的数据集巨大，而所要关心的区域只占网络的一部分，所以我们将采用划分式聚类来处理数据。在第三章中，将详细介绍我们采用的Betweenness聚类算法以及我们利用Modularity度量进行自动化改进的B-M算法。

## 团伙演化临界事件侦测

社会中人与人之间的关系时时刻刻都在发生着变化，同样地，在我们抽象出来的社会网络的结构和团伙构成也在时时刻刻发生着变化。研究随时间变化社会网络的演化规律对帮助商家把握流行规律制定有效营销策略，新闻学把握舆论动态，传染病学把握疾病传播的规律制定有效防疫措施等等应用都有着积极的意义。

我们将在第四章中应用临界事件的方式来描述社会网络的这种动态演化过程。所谓临界事件就是网络中的元素（个体和团伙）结构或状态发生重大变化的事件。例如节点的常见临界事件有：出现、消失、进入团伙、离开团伙；团伙的常见临界事件有：生成、分解、延续、合并、分裂等。定量地定义和发掘网络演化过程中的临界事件有助于我们科学地理解网络的动态演化规律。第四章将给出一些基本临界事件的定量定义以及侦测方法。

团伙发现算法

## 背景概述与问题描述

社会网络中的团伙是指一群具有相同兴趣爱好、有共同目的、从事相同或相似工作的人等。按照研究的角度不同，按照不同的共性将相似的个体聚集到一起，就能发现社会网络中的团伙。这就需要用到聚类技术（算法）。聚类是数据挖掘领域的重要课题，已经有数十年的研究历史，有很多成熟的算法，如层次聚类、k-means聚类等等。但是，面对纷繁复杂的数据集和各种各样的应用场景，没有哪种聚类能够通用于所有实际应用中。在我们的研究目标——基于通话记录的团伙发现应用背景中（将在第4章详细介绍），我们面对巨大的网络，但只关心其中含有指定节点的一个团伙，因此，我们选择了一种典型的切片聚类算法（通过去除网络图中的边达到聚类效果）——Betweenness聚类算法，这将在第二节中详细介绍。此外，为了达到无人监视的自动化团伙发现，我们引入了Modularity度量，在此基础上提出了B-M自动聚类的算法，将在第三节详细介绍。我们还将在第四节中设计多个实验验证该算法效率和准确性。最后，针对我们的应用场景，我们在第五节提出了一种扩散-剪枝的算法思路。

## Betweenness算法

M. E. J. Newman and M. Girvan在其经典论文[1]中提出经典的betweenness算法。所谓betweenness，就是最短路径betweenness，这是边上的一个权重度量（当然后续的研究者也在节点上设置betweenness度量）。分析网络中所有点两两之间的最短路径，统计通过某边Ei的最短路径条数b，就称作这条边上的betweenness。而最短路径通量——betweenness最高的边就是网络中的关键路径，这些边是连接团伙间的桥梁。将这些关键路径“切”去，就可以发现网络中的团伙。

Betweenness聚类算法的流程如下：

1. 计算图中各边的betweenness
2. 找出betweenness最高的边(如果包含多条相等最大的边就随机选一条)，并在团伙中去掉这条边
3. 重新计算网络的betweenness
4. 从步骤2)循环计算

每次循环切去一条边，切去指定的边数，或者团伙情况达到预期结果就停止计算。

可以看出，本算法的关键在于betweenness的定义和计算上。下面，我们以树形的网络结构为例叙述betweenness的定义，如图3-1(a)所示。边上的数字就是从s点出发到各点的最短路径条数。对于一些复杂的图，两点间的最短路径可能不止一条，如图3-1 (b)所示。节点边的数字代表从s点出发到该点的最短路径条数，边上的数字代表该边上的betweenness。对于最短路径有k条的两节点，每条最短路径分流得到1/k的betweenness，叠加计算到通过的边上。

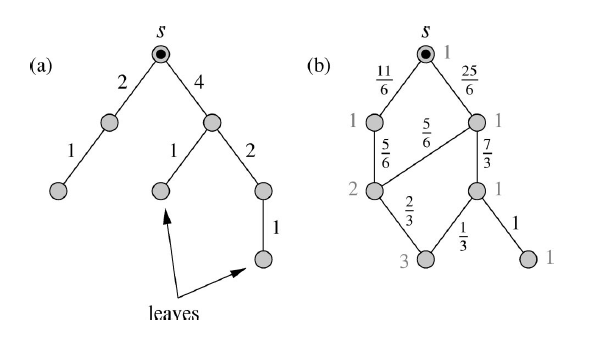


图3- 1 betwenness算法示意图

上述即是利用定义法求Betweenness的算法。但使用定义法求Betweenness效率并不高。设再网络图G中有n个节点(Vertex)，m条边(Edge)。两两计算任意2个节点间的最短路径，这样的计算需要进行次。而从一个节点出发做一次宽度优先遍历发现最短路径的最佳时间复杂性是O(m)。因此定义法求Betweenness的算法时间复杂度为O(n3)。空间复杂度为O(n2)，考虑存储图信息的矩阵和最短路径计算过程的消耗。

Ulrik Brandes曾提出一个优化的betweenness算法[8]，其计算空间复杂度为O(m+n)，时间复杂度为无权图O(mn)或有权图O(mn+n2log n)。在改进的betweenness算法中，Ulrik引入节点权重记录最短路径流，运用动态规划的思想，利用已经计算完成的路径的关键度权重来支持后面的计算，减少了重复计算，提高算法效率。如图3-2所示，三个阴影的三角区域是已经计算好betweenness的图区域，根据w1,w2,w3上承载的betweenness流量和v-wi边的betweenness可以递归地计算出sv上的最短路径。算法关注和更新每个节点上的betweenness权重，在此同时，可以同时维护更新每条边的betweenness权重，如图3-2所示，在更新节点v的betweenness的同时，vw1, vw2, vw3作为v的入边，承载三个区域增量的betweenness值。节点betweenness计算完成时，各条边的betweenness也同时得出。

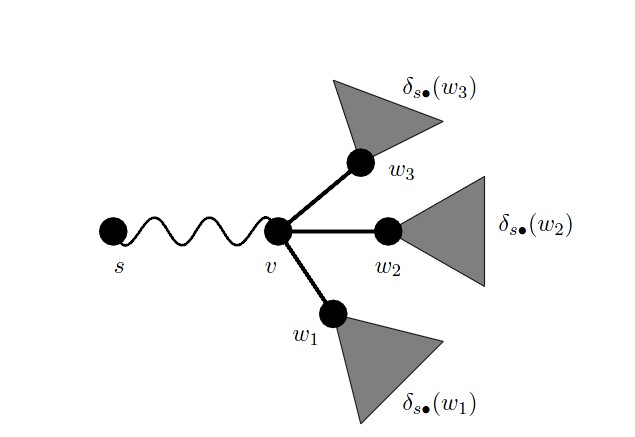


图3- 2 Ulrik的快速Betweenness计算算法示意

如算法3-1所示，即是Ulrik的快速betweenness算法伪代码。

|  |
| --- |
| 算法3-1 Ulrik的快速betweenness算法伪代码  设，在待研究图G中，V是其点集，E是其边集  BV[v]是各节点的betweenness权重，BE[e]是各边的betweenness权重 |
| BV[v] 🡨 0, v∈V;  BE[e]🡨0,e∈E;  **for** s∈V **do**  s 🡨 empty stack  P[w]🡨empty list, w∈V;  σ[t]🡨0, t∈V;  σ[s]🡨1;  d[t]🡨 -1, t∈V;  d[s]🡨 0  Q🡨empty queue;  enqueue s 🡪 Q  **while** Q not empty **do**  dequeue v 🡨 Q  push v🡪S  **foreach** neighbor w **of** v **do**  **if** d[w] < 0 **then**  enqueue w 🡪 Q  d[w]🡨d[v]+1  **end**  **if** d[w]=d[v]+1 **then**  σ[w]🡨 σ[w] + σ[v]  append v 🡪 P[w]  **end**  **end**  **end**  δ[v]🡨0,v∈V  **while** S not empty **do**  pop w🡨S  **for** v∈P[w] **do**  addition =σ[v] / σ[w] \* (1 +δ[w])  δ[v]🡨 δ[v] + addition  **foreach** incoming\_edge e **of** v **do**  BE[e]🡨 BE[e] + addition  **end**  **end**  **if** w ≠ s **then**  BV[v]🡨 BV[v] + δ[w]  **end**  **end**  **end** |

算法3- 1 Ulrik的快速betweenness算法伪代码

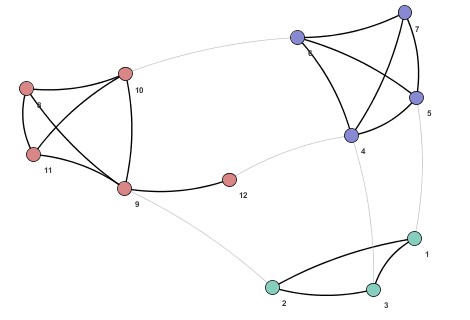


图3- 3 Betweenness切片聚类示意

如图3-3所示，是在我编写的程序中一次betweenness聚类的GUI界面图示。其中灰色的边表示被切取的边，去掉一定边数之后，网络图分成了三个团伙，不同的节点颜色代表不同的团伙。

## Modularity度量与自动聚类

Modularity度量

M. E. J. Newman and M. Girvan的论文[1]中还提出了Modularity的定义。关于Modularity的详尽定义在他们的另一篇论文[2]中被提出。Modularity通过考察网络的结构和聚类的情况来度量该次聚类的准确性。Modularity计算团伙内和团伙间的连接占网络总体连接的比例，从一定侧面反映聚类的质量。主要有以下几种计算方式。

Modularity的原始定义如公式3-1所示。在一个有m条边，n个顶点的图中，矩阵A是图的邻接矩阵，设ki是节点i的度数，ci是节点i所在的团伙。δ(ci,cj)是kronecker delta，即克罗内克符号函数，那么modularity的计算公式如下：

公式3-1

Newman在论文[1]中提到的一个常用Modularity计算方式如公式3-2所示。一个网络被分为k个团伙，那么我们设置一个k×k的矩阵e，其中的元素eij表示，团伙i到团伙j的连接边占网络总边数的比例，再设ai是矩阵上第i行的元素和（也可求列和）。记Tr **e** 为矩阵**e**的迹，即主对角线的和。记||**e**||为矩阵的元素和。那么modularity度量的计算公式的一种表达如下：

公式3-2

Modularity的计算有多种推论公式，其中在Ulrik Brandes等人的论文[12]中就提出多种推导形式的Modularity计算公式。其中一种比较简洁的形式如公式3-3所示，该计算方法比较适宜编写程序计算，也被我用在我的程序中。其中m为网络中边的总条数，k为团伙个数，mi为团伙i内部边的条数，di为团伙i内各点的度数和：

公式3-3

利用modularity衡量聚类的效果。如图3-4所示，对于同一个网络，不同的聚类结果具有不同的Modularity值。在(a)中，modularity几乎是0，在(b)中modularity的值接近于0.5。

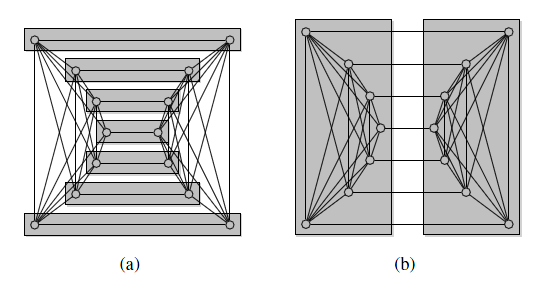


图3- 4 对于同一个网络的不同聚类效果， Modularity度量的效果示意

Modularity 的值在-1.0~1.0之间，在实际的网络图中，比较好的聚类结果，modularity的值通常在0.3~0.7左右，较高的modularity值是很罕见的。

利用Modularity进行自动聚类

在使用基本的betweenness聚类算法进行团伙发现时，需要指定切去的边的条数来界定切边是否停止。这样，在实际应用中就需要监视聚类的情况，这在实际生产中是很难实现的。在引入Modularity度量后，我们可以发现，逐一切取网络中的边，Modularity度量值会呈一定规律地变化，如图3-5所示。

图3- 5 逐一切边与modularity的变化

随着切边的进行，我们考察网络的modularity度量值的变化，可以发现度量值会呈现一个或多个峰值。图3-5中是比较常见的2个例子。大多情况下（七成左右），modularity呈现一个峰值。某些数据下，也可能如case2一样出现多个峰值。Modularity取得较高的值，即说明取得了比较好的聚类效果。

我们也可以注意到，切去几条边呈现的modularity值可能是一样的。例如case1中的切取5,6,7三条边所呈现的modularity是相同的。这是因为modularity度量是考察团伙聚集情况的，而一般来说，需要切取多条团伙间的联系，才能使团伙分割开。可以参看图3-4中红色和蓝色的2个团，单切去边6-10聚类情况并不改变，modularity值也不变；只有继续切取变4-12之后，红色和蓝色团才分开，modularity才会发生变化。

根据上面提到的modularity分布情况，我们可以很自然地想到，逐一切去一个网络图中的所有边，同时记录切边过程中modularity的值。最后取定modularity最大的时候团伙的情况作为我们聚类的结果。这样就可以实现自动地（无监视）地对网络进行聚类分析了。这就是完全地betweenness-modularity自动聚类（以下成完全的B-M自动聚类），算法流程如下：

1. 度量网络的betweenness
2. 切去betweenness最大的边（关键路径）
3. 如果团伙情况发生变化，计算modularity值
4. 记录切边轨迹
5. 如果网络中还有边，重复步骤1）
6. 考察切边轨迹，取modularity最大的情况，恢复图，得出聚类结果

这样能够保证得出modularity最优的聚类情况，但是算法的开销代价也是巨大的。每次切边都要度量betweenness值，而计算betweenness的算法复杂度是O(mn)，（参看本文第3章第1节）。切去所有的m条边，切边的操作代价是O(m)，计算每次betweenness的代价是O(m2n)。网络图从1个团（未处理），到n个团（所有点都独立），需要n次计算modularity的值，即需要O(n)的开销。综上，完全的B-M自动聚类的时间复杂度为O(n3)。

面对大数据集的处理，我们也可以使用一种贪心的策略来自动聚类。即选取modularity的第一个峰值，就停止迭代计算。算法流程如下：

1. 度量网络的betweenness
2. 切去betweenness最大的边（关键路径）
3. 如果团伙情况发生变化，计算modularity值
4. 记录切边轨迹
5. 考察切边轨迹，如果呈现峰值（第一次峰值）迭代停止，进入第6)步；否则重复步骤1)
6. 恢复图的第一峰值时的情况，得出聚类结果

使用贪心的B-M自动聚类法可以将算法效率在较好的情况下（峰值出现早）提高到O(n2)，最差的情况下（峰值最后才出现）退化为完全的B-M自动聚类，即O(n3)。通常情况下，第一峰值的效果就能达到较好的效果，但贪心法不可能保证所有情况都能得出最优的聚类效果。具体的算法效果比较见3.2.3的实验设计与结果分析。

## 基于特定团伙的分析——扩散-剪枝算法框架

在实际的生产应用中，通常要面对的数据集对象都非常庞大，例如本文研究的一个实例数据集（255个节点）中，就有460,369条通话记录。这些记录是一年来255个节点产生的通话记录，由于隐私权限制，我没有取得全量的数据。我通过实验将这个数据集进行了扩增和模拟(扩增与这些节点有关的点)，产生的数据集有3,264个节点，72,354,329条数据。工业和信息化部最新数据显示，截至2011年4月底我国移动电话用户数突破9亿户。这么多节点之间产生的通话记录数据将是惊人和难以计算的。

然而，在实际应用中，我们需要关心的只是这张巨大网络的一个局部。例如，公安机关利用通话记录辅助破获犯罪团伙时，与犯罪团伙有关联的人员只占整个网络的一小部分。而在最初，一般都会已知团伙中的一个或多个犯罪嫌疑人。因此，在实际应用中，我们只需要从已知的节点出发，扩展与这些已知节点相关的网络。然后再对这个子网进行分析，剪枝去掉相关性较弱的节点，就可以得出我们最感兴趣的团伙情况。

如图3-6所示，我们所关心的团伙，在全网中只是一个很小的部分，图中灰色区域表示全网，淡黄色部分是我们所关心的子网，包含已知的3个红色的关键点。对全网进行分析开销是巨大的，我们只需要对所关心的那个小区域进行分析就足够了。

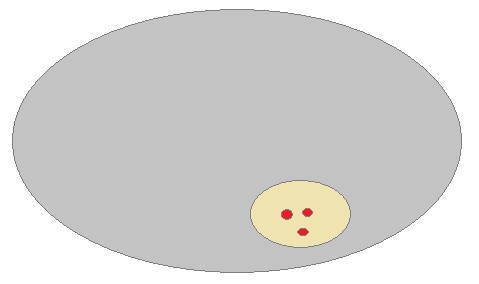


图3- 6 图示，我们所关心的网络局部只占全网的一部分

扩散-剪枝算法框架流程：

1. 从已知点集V出发，扩展与V有关联的所有节点，并加入到集合V中
2. 迭代步骤1)直到扩展得到的子网Gs符合研究的需要（一般2~3次即可）
3. 对Gs进行聚类分析，切去关联度不大的节点
4. 得出我们所关心的团伙

根据六度空间理论，任何两个陌生人之间所间隔的人不会超过六个。我们进行的扩展并不需要进行很多次，否则就会得到过于庞大的子网，不利于下一步的分析。通常来说，扩展2~3次就足够了。实际上，与关键节点距离超过3的点与我们所关心的团伙联系已经非常疏远了。

对于扩展的算法是比较自然的，只要从已知关键点出发，进行宽度优先遍历，扩展与关键点有关联的网络空间即可。

对于对上一步挖掘出的子网进行进一步剪枝分析的算法选择，我们可以选择具有剪枝特性的切边或切点的算法作为聚类技术是比较适宜的。例如本文的研究中就选取B-M自动聚类算法来对子网进行聚类分析。因为在聚类的过程中，对于我们并不关心的团伙，我们就没比较再对它们进行迭代计算，从而进一步降低计算的开销。如图3-7所示，我们对关心的淡黄色团伙进行迭代剪枝分析，而灰色区别则无需再进一步计算。逐渐剪去与关注团伙关联度小的节点，最终就得到我们所关注的团伙情况了。

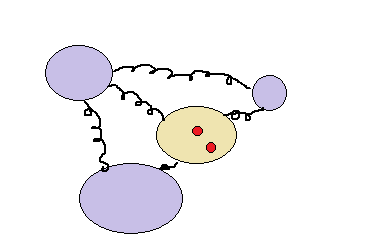


图3- 7 剪枝过程剪去不需关注的与关键团伙疏远的边

本文将会在第4章，通过一个实例来进一步说明扩散-剪枝算法框架的实际应用。

## 实验设计与分析

我们的实验数据是实验模拟生成的网络数据。数据生成器的算法来自于Newman的论文[1]，这个数据生成算法也被其他研究者所使用，如[13]，是一个常用的数据生成方法。我们用java实现了这个算法，并生成数据用于实验。

数据生成器接受以下的关键指标以生成实验数据：

State：随机数源的状态，一般不用指定

size：数据集的节点数量，例如128

avgDegree：每个节点的平均度数，例如16

z：每个节点上连接其他团伙的平均度数，例如2。该参数用于调整干扰因子的大小。

我们定义干扰因子p = z/avgDegree来描述数据集内的干扰因素。如上面的例子中p=2/16=12.5%，就说明网络中有大约12.5%的边是连接不同团伙的，即又12.5%的干扰数据。

在实验中，我们使用NMI值来衡量聚类的准确度。NMI是反映聚类结果与网络真实分类情况的一个度量。首次被A. Strehl 和 J. Ghosh在其经典论文[14]中提出。计算公式如下：

（其中I(X,Y)表示X与Y之间的互信息，H(X)表示X的熵，H(X)=I(X,X) ）

在我们的场景中，互信息I(X,Y)可以按如下公式计算：

（其中表示X中第h个团伙与Y中第l个团伙的交集节点数，n表示总的节点数，表示X中第h个团伙中的节点数）

实验用代码使用Java实现。系统环境如表3-1所示：

|  |  |
| --- | --- |
| 操作系统 | windows 7 旗舰版 |
| CPU | Intel(R) Genuine T2080 双核1.73GHz |
| 内存(RAM) | 2.50GB |
| JVM版本 | 1.6.0\_12 |

表3- 1 实验环境

实验1：聚类准确性测试

本实验旨在测试B-M算法的聚类准确度，控制数据生成器所生成的数据规模不变，改变干扰因子z的值，分别测试贪心法和完全法达到的效果，使用NMI来衡量聚类准确性。

#### 实验数据：

|  |  |  |
| --- | --- | --- |
| 变量 | 描述 | 取值范围 |
| State | 随机数生成器状态 | 10 |
| size | 数据的节点数 | 128 |
| avgDegree | 每个节点的平均度数 | 16 |
| z | 每个节点的干扰度数 | 0~8 |
| P | 干扰因子 | 0%~50% |

表3- 2 实验1数据说明

#### 实验结果：

图3- 6 实验1实验结果

#### 结果分析：

通常来说，NMI的值在0.6以上，我们可以认为聚类的质量是可以接受的。如图3-6所示，我们可以看出，B-M算法在使用贪心法时在干扰因子z<3,即干扰比p<18.75%以下时可以产生较好的聚类效果；完全法可以承受干扰因子z<5,即干扰比p<31.25%。

贪心法比完全法较早地出现了不稳定的情况，这是由于贪心法只取第一个modularity峰值。当网络聚集度较高时，modularity呈现较为规律的峰值分布，可参看图3-5 case1，贪心法可以取得较好的聚类效果；而当网络干扰增大，聚集度降低时，modularity曲线呈现较为混乱的峰值分布，可参看图3-5 case2，贪心法就难以保证取得最佳的效果了。

实验2：聚类效率测试

本实验旨在测试B-M算法的聚类效率，控制数据生成器所生成的数据秩序不变，改变数据规模size的值，分别测试贪心法和完全法达到的效果。

#### 实验数据：

|  |  |  |
| --- | --- | --- |
| 变量 | 描述 | 取值范围 |
| State | 随机数生成器状态 | 10 |
| size | 数据的节点数 | 16~128 |
| avgDegree | 每个节点的平均度数 | 16 |
| z | 每个节点的干扰度数 | 2 |
| P | 干扰因子 | 12.5% |

表3- 3 实验2数据说明

#### 实验结果：

我们使用JDK自带的jconsole工具监视聚类过程中内存和CPU的消耗，如图3-7和图3-8所示。在不同的数据规模下，我们度量聚类所需的时间，如图3-9所示。不同数据规模下的CPU平均消耗如图3-10所示，内存消耗如图3-11所示。

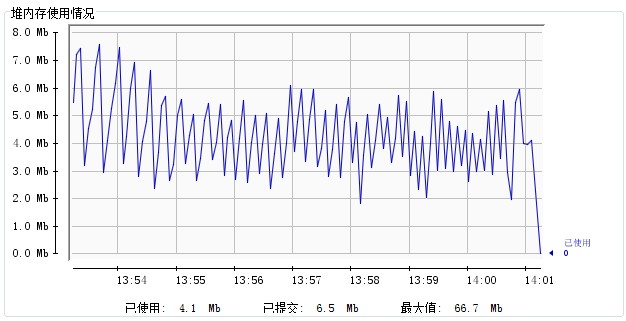


图3- 7实验2聚类过程中内存的消耗

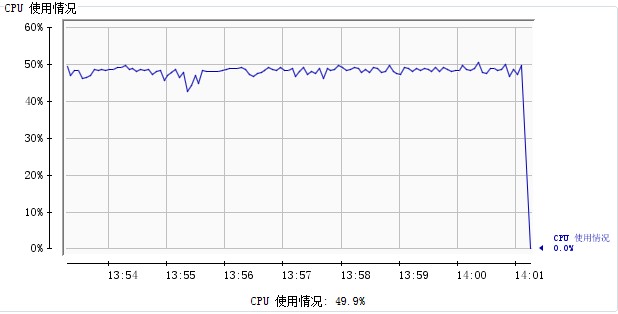


图3- 8实验2聚类过程中CPU的消耗

图3- 9实验2效率测试的结果，时间消耗

图3- 10实验2效率测试的结果，平均CPU消耗

图3- 11实验2效率测试的结果，平均内存消耗

#### 结果分析：

由B-M聚类的原理可知，在聚类进行的过程中，并不会反复申请额外的空间，主要的消耗是存储网络结构和中间结果。图3-7的结果也证明了这点，堆内存消耗在均值周围规律波动，波动的原因和JVM的内存回收机制有关，不会出现消耗激增的情况，内存消耗稳定。

CPU的消耗和JVM的环境设定有关，在我的实验环境里，CPU用量维持在50%左右，图3-8和图3-10中的结果反映了这点。

再看聚类的时间消耗，完全B-M聚类的时间复杂度是O(n3)，从图3-9也可以观察到，随着数据规模的增大，时间的消耗增加很快，符合算法复杂度分析。另外可以看到，贪心法B-M聚类的时间消耗虽然保持和完全聚类相似的增长度，但波动比较大。这是因为贪心法聚类时，出现第一个betweenness峰值的时间具有一定的偶然性。关于贪心法比完全法节省的时间，根据不同的数据集，有较大的偶然性和波动性。根据我们其他更多的实验显示（未在图中展示），贪心法平均能比完全法节省35~60%的时间。

在内存消耗上，贪心法和完全法区别不大。因为策略虽然不同，但是都需要存储图的结构和存储中间变量，内存消耗相近，如图3-11所示。

本章小结

本章第一节详细介绍了使用betweenness切边聚类算法对网络图进行聚类分析并发现团伙的算法。重点介绍了Ulrik在[8]中提出的快速betweenness计算算法。

本章第二节从度量聚类效果的modularity定义和计算讲起，提出一种利用modularity度量进行自动聚类的方法——B-M自动聚类算法。最后还对该自动聚类算法进行了实验验证。经过实验验证，可以看出，随着网络规模的增大，算法的时间消耗呈现如同分析中提到的O(n2)到O(n3)的增长，符合预期期望。其中，贪心法B-M自动聚类算法比起完全的B-M自动聚类能够非常快速的得出结果。在聚类准确性验证方面，随着网络中干扰因子的增大，在一定范围内，聚类的效果都比较理想和稳定。干扰因子较大的情况下和某些case下，贪心法B-M聚类的效果会出现不稳定和不理想的状况。

本章第三节还依据我们现有的应用场景，提出一种扩散-剪枝的算法框架，或者说是算法思路。在实际应用中，避免搜索全网，对网络局部进行分析，降低计算的消耗。

　团伙演化的临界事件侦测算法

背景概述与问题描述

研究背景和问题描述

在实际的社会网络中，人与人之间的关系随着时间的推移，时时刻刻在发生着改变。由人际关系抽象出来的社会网络图的结构也时时刻刻发生着改变。在这个变化的社会网络中，根据相同的兴趣爱好或者共同的目的“聚集”在一起的团伙也时时刻刻在发生着诸如分裂、合并、生长、消亡这样的变化。研究团伙随着时间推移的动态变化规律有着非常重要的现实意义。在社会网络中，研究团伙的变化规律可以帮助商家把握市场的动态，从而制定出有效的营销策略；可以帮助社会学家研究社会的演化规律。在预防学领域，可以帮助传染病专家研究传染病的传播规律和趋势，制定有效的预防措施。在生物学领域，可以帮助生物学家研究种群的演化规律。等等。

研究以时间维度（其他维度也可类推）为基线变化的动态网络，一个最行之有效的方法就是对动态网络在各个时间点上的状态做切片，然后分析各切片之间的演化情况。

所谓切片，并不是时刻点上的横断面，它是一个小时间区间Δt上的网络关系聚合。例如对于基于通话记录的网络图切片，切片t1是用户1~3月的通话记录建模构建的网络图，切片t2是用户4~6月的通话记录建模构建的网络图。t1与t2的叠加即为根据用户1~6月通话记录建模后构建的网络图。图4-1所示就是3个切片的示意图。T1是在时刻t=1时的网络图结构;T2是t=2时刻的结构。T1,2是T1时间片和T2时间片的聚合。



图4- 1时间切片的概念，从左到右分别是：a)时间切片t=1；b)时间切片t=2；c)时间累积切片t=1,2. 这是t1时间片和t2时间片的关系叠加

随着时间的推移，网络图时刻在发生着演化。如图4-2所示，T1~T5五个时间片上，网络的结构各不相同。随着时间的推移，网络中的团伙也在发生演化，图中用相同颜色来标注相同的团伙。

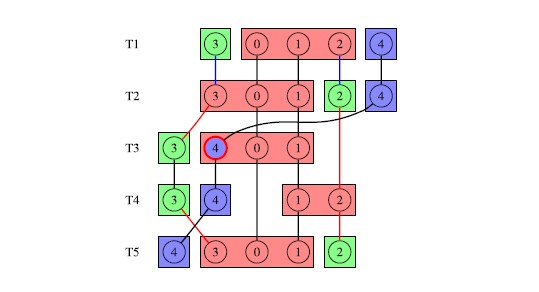


图4- 2 随着时间的推移，图的结构发生变化的示意图。图中Ti是时间片序列。随着时间的变化，图的结构发生着变化。展示了5个时间片上不同的图结构，以及团伙的保持和进化

网络图中有两类关键实体，一个是个体（个人），另一个是团伙。对于个体的行为事件，比较简单，大致有四种基本事件：1.出现，2.消失，3.进入团伙，4离开团伙[3]。团伙的演化行为则比较复杂，大致可以分为一下几个基本事件，如图4-3所示：1. 团伙成长；2. 团伙衰退；3. 团伙合并；4. 团伙分裂；5. 团伙出现；6. 团伙瓦解。

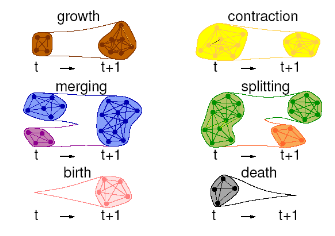


图4- 3 团伙的基本事件

下面，我们就重点讨论一下临界事件的定义。

个人临界事件的定义

#### 出现Appear

一个在时间片T1上没有的点A，出现在时间片T2上，我们就说A发生了出现事件。设T1上的点集为V1，T2上的点集为V2，有如下数学关系：

#### 消失Disappear

一个在时间片T1上的点D，在时间片T2上消失了，我们就说D发生了消失事件。设T1上的点集为V1，T2上的点集为V2，有如下数学关系：

#### 进入团伙Join

一个时间片T1上不属于团伙的点V，在T2时间片属于团伙，我们就说这个节点发生了进入事件。满足进入事件的点有如下数学关系

#### 离开团伙Leave

一个时间片T1上属于团伙的点V，在T2时间片不属于团伙，我们就说这个节点发生了离开事件。满足离开事件的点有如下数学关系

团伙临界事件的定义

团伙的演化是连续的，为了标定团伙演化的关键性时刻，Asur等人在[3]中首次定量地提出了临界事件的概念。通过侦测临界事件，就可以定量地研究团伙的演化过程。在Asur的研究中，定义了5中基本的临界事件：

设Si和Si+1为时间片集合S上的两个连续时间片，它们中包含的团伙集合分别为Ci和Ci+1。以下是这五种基本事件的定义：

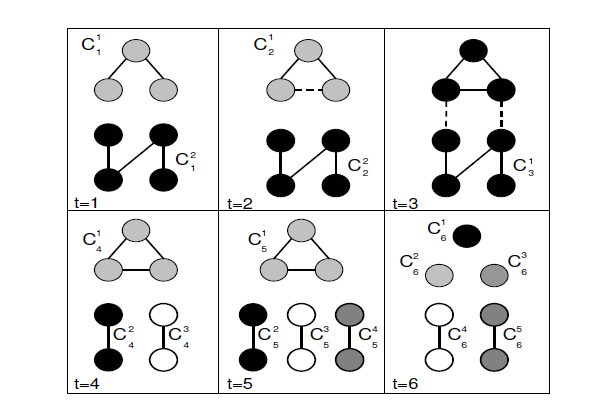


图4- 4一个网络图在6个时间片t=1到6上的时间片。不同的团伙使用不同的颜色标出。下文用于参考临界事件的示意

#### 延续Continue

当团伙内的点集与上的点集相同时，我们说这个团伙发生延续事件。我们不需要关心边的变化：

提出这个定义的动机是：如果一个团伙的成员一直保持，那么团伙中任一点的信息都可以最终到达另外任一点。也就是说，只要团伙点集保持不变，那么信息流不会收到影响。边的增减只对团内节点的关联度产生影响。如图4- 4所示t=1和t=2时间片上，两个团伙都发生延续事件。特别注意到，在上虽然增加了一条边(联系)，但团伙并没有发生改变。

#### K-合并k-Merge

两个不同的团伙和发生k-合并事件，当且仅当在下一时间片上，存在一个团伙至少保有了这两个团伙里k%的节点。必要条件就是:

满足

并且并且

这种情况下，在时间片i+1上，和之间必然是存在着边(联系)的。直观地说，合并表达的就是在原先两个团伙之间形成了新的联系。这使得原先2个团伙中k%的成员构成了新的团伙。可以看出，存在着一种完全的合并事件，也就是当k=100时，两个团伙的成员完全合并到下一时间片上的一个团伙中。在这种情况下，原有的两个团伙完全消失。如图4- 4所示，t=3时就发生了完全合并。虚线指示的边新创建出来，所有的节点都合并到中。

#### k-分裂k-Split

某个团发生了分裂事件，当这个团中k%的成员在下一时间片分裂到两个不同的团伙中。其必要条件为：

满足

并且 并且

直观上来说，就是这一时间片上相关节点间的联系在下一时间片上被打断了，导致这些节点在下一时间片加入(组成)两个团。需要注意到，一个边的断开，并不一定意味着发生了分裂事件，因为还可能存在其他的联系保持了这个团伙的构成（和k-连通性的概念类似）。在图4- 4中，t=4时刻就发生一次完全的分裂事件，原团完全分裂为3个团伙。

#### 生成Form

一个新的团伙发生了生成事件，当且仅当这个团伙的任意两个节点在前面的时间片中不同属于任一团伙中。或者说，没有任何两个节点在是在t=i时间片上的同一团伙的。

直观地说，生成代表着一个新团伙的建立。在图4- 4中，t=5时刻就出现了2个以前没有的新节点，并发生了团伙的生成事件。

#### 分解Dissolve

一个团伙发生了分解事件，当且仅当在下一时间片上，这个团伙中的任意两个节点都不同属于任一团伙。或者说，起始团伙中的任意两个节点间都不将存在联系。

直观地说，在某一时间点上，一个团伙内部缺乏足够的联系，那么这个团伙就发生分解了。这意味这一个团伙(团队)的解体。图4- 4中t=6上指示了一次分解事件，当团伙中的三个节点不再有联系时，这个团伙分解成3个团伙

Asur等人定义的五种基本临界事件是比较严格的，有些时候，团伙的演化并没有严格地处在这五种事件中。例如一个延续的，相对稳定的团伙，总有旧的成员离开和新的成员加入。这样的情况并不严格遵守上述的Continue事件。从Asur五种基本事件出发的挖掘系统也不会捕捉到这样的事件。为了更详细地发现团伙演化的情况，我们在Asur五种基本事件的基础上增补定义了以下三种基本事件：

#### k-延续k-Continue

与Continue事件类似，当团伙内的点集包含上的点集上k%的个体的时候，我们说这个团伙发生k-延续事件。我们不需要关心边的变化：

增加这个事件的动机是为了更广泛地描述和发现延续的团伙。并且，我们认为，当一个团伙的规模扩大一倍以上或者缩小为以前的一半以下时，考察这种延续对实际应用的意义不大。因为在这些情况下团伙结构已经发生了巨大的变化，已经不适宜用延续来描述团伙的演化了。

#### k-成长k- Grow

当团伙内的点集规模扩大为上的点集规模的1+k%的时候，我们说这个团伙发生k-成长事件。

增加k-成长的动因在于描述处于增长阶段的团伙。另外，当一个团伙的的成员变化超过一半的时候，团伙的结构已经发生了重大转变，团伙的属性已经改变，不适宜使用增长来描述团伙的行为了。

#### k-衰退k- Decline

当团伙内的点集规模缩小为上的点集规模的k%的时候，我们说这个团伙发生k-衰退事件。

增加k-成长的动因在于描述处于衰退阶段的团伙。同样地，当一个团伙的的成员变化超过一半的时候，团伙的结构已经发生了重大转变，团伙的属性已经改变，不适宜使用衰退来描述团伙的行为了。这也要求衰退因子k在50以下。

上述临界事件中的2.3.6.7.8中的附加条件是为了使判定更为准确，可以根据实际情况做进一步调整。

临界事件侦测算法

在第一节中，我们已经给出了八种基本的团伙临界事件的定量定义。但无论是个人行为发现还是团伙演化研究，其中一个基本点也是难点在于切片之间团体的对应关系的发现和保持[7]。简单解释就是在切片t上的团体A0，到了切片t+1上，我们要在聚类出的团体集合中找到和A0对应的团体A1。如果发生的是分裂和合并还需要追踪到最为相近的一系列团伙。追踪的过程比较复杂也是动态挖掘的难点所在。研究的思路还是要化纷繁为简单，抓住核心。首先最基础的一个基线是找到最相近的团体。

要找到相近团体，最基本的是相似度的度量，我们使用Jaccard系数[6]进行度量。度量公式为：C1,C2的相似度 = |C1 ∪ C2| / |C1 ∪ C2|

切片t中团体集合St0={P0,P1,P2,…,Pn}，切片t+1中团体集合St1={Q0,Q1,Q2,…,Qm}，两两之间(Pi与Qj间)都会有一个相似度Jaccard系数。对于单团，取出最大的相似度对应团是非常自然的想法。但是不能这样简单地处理，必须全局地看待这个问题。即取出全局上最优的匹配组合。这样，这个问题可以转化为二部图的最佳匹配问题。对于二部图的最佳匹配问题，我将采用Kuhn－Munkras算法[10]来解决二部图匹配问题。关于KM算法的具体细节，这里就不赘述了。

完成了团伙的匹配查找之后，我们就可以根据团伙和个人的基本事件动作定义来发现两个时间片之间的临界事件了。

算法流程如下：

1. 在动态数据集上分割时间片（根据应用的实际需要分割和调整，例如我的应用中使用一个月的间隔作为分割时间片的依据）
2. 对相邻的两个时间片分别进行聚类分析
3. 使用KM算法匹配这两个作为研究对象的时间片上的团伙
4. 逐一检测每个节点，发现节点临界事件
5. 逐一检测每个团伙配对，发现团伙临界事件
6. 循环检测其他的时间片

显然地，如果不需要检测节点的行为，步骤4就可以省略；同样，如果不需要检测团伙的行为，步骤5就可以省略。

对于检测个人行为算法，出现和消失都是比较显而易见的现象，直接查询存储的数据即知。而在时间轴上完成了对团伙的追踪后，只需要检测个人所属的团伙id也可方便地知道个人的进入团伙和离开团伙行为。

在匹配完成后，根据定义检测团伙临界事件也不复杂。分成3类进行检测：（设两个相邻的时间片为T1,T2）

1. 未匹配的团伙。T1中未匹配的团伙D可能是在T2中分解的团伙，按照分裂的定量公式考察D的成员流向判定是否分裂。同理，T2中为匹配的团伙F可能是在T2中新生成的团伙，按照生成的定量公式考察F的成员流向判定是否发生生成事件
2. 匹配的团伙对，检测一对一的事件。一对一事件包含①延续⑥k延续⑦k成长⑧k衰退。按照各自的定量公式检测这对配对的团伙即可判定是否发生了临界事件
3. 匹配的团伙，检测一对多事件。一对多事件包含②k合并③k分裂。对于合并事件，需要分析与T2时间片的上团伙M最相近的两个T1时间片上的团伙M1和M2，利用定量公式判定它们之间是否发生了合并事件；同理，对于分裂事件，需要考察T1时间片上的团伙S及其在T2时间片上最相近的两个团伙S1和S2，利用定量公式判定它们之间是否发生了分裂事件。

至此，就完成了检测临界事件的流程，下面将通过一些实验来验证我们的理论。

实验设计与分析

本节实验中采用的实验数据来自于模拟数据生成器，该生成器在本文的3.4中有使用与介绍。在本节实验中，生成器作了一些增强，除了接受数据规模和干扰因子外，还需要接受生成事件的类型。每次生成连续2个切片的数据集，用于发现预先设定临界事件。

实验用代码使用Java实现。系统环境如表3-1所示：

|  |  |
| --- | --- |
| 操作系统 | windows 7 旗舰版 |
| CPU | Intel(R) Genuine T2080 双核1.73GHz |
| 内存(RAM) | 2.50GB |
| JVM版本 | 1.6.0\_12 |

表4-1 实验环境

### 实验1 临界事件准确度测试

本实验用于验证临界事件侦测算法的准确性，控制数据的规模（节点数和节点平均度数）不变、实验事件数不变，改变干扰因子的大小，测试侦测到临界事件的准确度。

#### 实验数据：

|  |  |  |
| --- | --- | --- |
| 变量 | 描述 | 取值范围 |
| State | 随机数生成器状态 | 10 |
| size | 数据的节点数 | 128 |
| avgDegree | 每个节点的平均度数 | 16 |
| EventCount | 用于实验的事件数量 | 20 |
| z | 每个节点的干扰度数 | 0~8 |
| P | 干扰因子 | 0%~50% |

表4-2 实验1数据说明

#### 实验结果：

每组实验（每个干扰因子一组实验）都给出20个临界事件供事件侦测算法去发现，用命中(hit)数来衡量侦测算法的准确度。全命中为20次。

图4- 5实验1个人临界事件准确度测试

图4- 6实验1团伙临界事件准确度测试

#### 结果分析：

对于个人事件的侦测，如图4-5所示。出现和消失事件20次case全命中，因为出现和消失事件比较单纯，就是指是否在该切片上有这个点，而且该事件的模拟与我们的干扰因子无关，所以判定是完全准确的。侦测节点的进入和离开事件的准确度主要依赖于聚类的准确度。参看3.5.1的聚类准确度测试，我们可以发现，进入和离开事件的侦测准确率和聚类NMI效果非常吻合。

再看团伙事件的准确度，八种团伙事件的命中率也与聚类效果NMI度量基本吻合。在z小于4的情况下，我们侦测事件的准确度比较理想，基本能够命中预设定的场景。随着干扰因子的增大，团伙的聚集性降低，聚类的情况也与预设定情况发生偏差，临界事件发现的准确度也相对降低。

### 实验2 临界事件效率测试

本实验旨在检验临界事件侦测算法的效率，控制数据的干扰因子不变、实验事件数不变，改变数据规模的大小，测试侦测到临界事件的时间。

#### 实验数据：

|  |  |  |
| --- | --- | --- |
| 变量 | 描述 | 取值范围 |
| State | 随机数生成器状态 | 10 |
| size | 数据的节点数 | 16~128 |
| avgDegree | 每个节点的平均度数 | 16 |
| EventCount | 用于实验的事件数量 | 5 |
| z | 每个节点的干扰度数 | 2 |
| P | 干扰因子 | 12.5% |

表4-3 实验2数据说明

我们将使用JDK自带的jconsole工具分析我们的计算过程中对内存和CPU的消耗变化。还将比对不同数据规模下的时间和系统资源消耗量，同时拿该数据集的只聚类不侦测动态事件(cluster only)的资源消耗和完全分析(cluster & find events)的资源消耗做比较。

#### 实验结果：

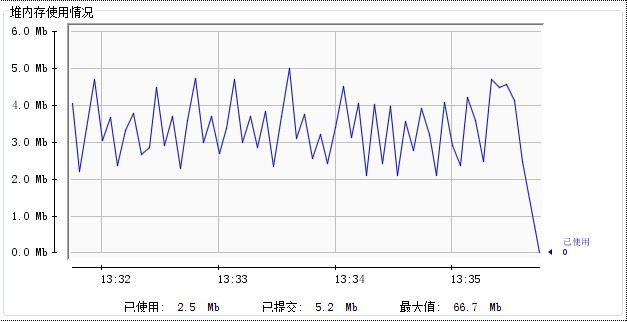


图4- 7实验2分析过程中内存的消耗



图4- 8实验2分析过程中CPU的消耗

图4- 9实验2效率测试的结果，时间消耗

图4- 10实验2效率测试的结果，平均CPU消耗

图4- 11实验2效率测试的结果，平均内存消耗

#### 结果分析：

根据前文对算法的分析，我们在计算过程中一直使用的是存储网络结构和中间变量的内存资源，不会因分析阶段的不同造成内存用量的大波动。实验也证明了这一点，如图4-7所示。规律性的波动是因JVM的垃圾回收机制引起的正常变化。

计算的CPU用量与JVM的环境配置有关，在我的实验中，始终维持在50%左右，如图4-8和图4-10所示。

需要注意的计算的时间消耗，如图4-9所示。可以看出，随着数据规模的增大，时间消耗增大非常明显，曲线符合前文算法分析中指出O(n3)的复杂度。另外，值得一提的是，我们的时间消耗主要来自于聚类计算过程，侦测事件的过程占总时间消耗的10%~20%之间。这是由于聚类计算所涉及的是网络中的所有节点，而事件发现的主要消耗是网络中的所有团伙；网络中节点的数量远多于团伙的数量。所以，聚类计算占用了总计算时间的大部分。

最后，我们再来看随着数据规模的增大内存的用量变化。内存主要用来存储网络的结构和随结构增长的中间变量。其用量呈现比较平缓的增长。

本章小结

社会网络随着时间的推进每时每刻都在发生着变化，研究包含着这种动态变化的网络及其中的团伙和个体的行为规律对我们有着重要的意义。本章在开头第一节介绍了研究动态网络的常规方法——利用时间切片和发掘关键事件来描述和分析。紧接着还介绍了个人和团伙的一些临界事件及其定义。在这一部分，本文除了介绍Asur提出的五种团伙基本临界事件外，还提出了三种增补的团伙临界事件，更为详尽地描述团伙在演化过程中发生的行为。

本章第二节重点介绍了怎样实现算法来发掘动态网络中的临界事件，主要涉及使用KM算法匹配团伙和使用定量公式判定临界事件。

本章第三节设计了一些场景和实验数据验证了我们的实现的算法。可以看出，在干扰因子较小的情况下，我们的判定命中率非常高。当干扰因子增大时，判定就会出现误差。出现误差的原因主要来自于聚类的结果与我们预设定场景的差异。另外在实验中也可以看出，计算的时间开销是非常大的，主要的计算消耗来自于聚类计算。而匹配和判定的消耗主要依赖于团伙个数的规模，团伙个数一般比节点个数少一个数量级，O(n2)的计算复杂度也比聚类的O(n3)复杂度要低，因此临界事件的判定计算消耗占比是不大的，主要消耗来自聚类。提高聚类的效率（优化聚类算法或者降低标准使用贪心算法）有助于提高算法的效率。另外，提取出实际应用关心的区域进行分析取代全网挖掘可以在实际应用大大降低计算的消耗。

一个基于通话记录的应用实例

背景概述与问题描述

社会网络中人与人之间的联系可以表现在很多方面，例如：个人之间的通话记录，微薄的互相关注，博客社区的相互引用和浏览，Email通信记录，共同参与某些活动等等。通过对这些典型的数据集进行分析可以帮助我们在一定程度上掌握社会网络的结构。

通话记录作为一种常见的能有效反应社会网络结构的数据集，常被用作分析对象。对通话记录数据样本的分析，发掘出其中的团伙及其变化规律可以为公安机关侦破团伙犯罪提供有力的线索。

由于隐私权限制，本文取得一个包含255个节点的通话记录样本，该样本包含这些节点在一年中产生的460,369条通话记录。通话记录并不能直接被用于分析，还要进行一定的建模和预处理，对通话记录的数据建模将在第二节详细描述。完成数学建模后，通话记录数据就可以抽象为一张巨大的有权网络图。由于发现团伙的聚类分析计算开销非常大，本章第三节将使用前文2.3提到的扩散-剪枝算法框架发现几个含有关键节点的典型团伙。最后，我们还将利用前文第三章中提到的定义和算法分析这个动态网络中临界事件，从而追踪团伙演化的规律。

基于通话数据的数学建模

原始的通话记录数据分散的，无法被用作分析，必须进行一定的数学建模，使其成为一个有权的网络图。通话记录数据集的结构如下：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 字段意义 | 主叫号码代号 | 通话开始时间 | 被叫号码代号 | 交换机编号 | 基站代码 | 通话时长（秒） |
| 示例数据样本 | 40159533210 | 2009-01-30 19:23:59 | 41611388610 | 377202 | 006d3 | 43 |

号码代号是为了保护用户隐私对手机号码进行的一次线性加密编码，可以认为是唯一标识个人的id。除了双方代号，我们还需要关注通话的时刻，即通话开始的时间，这个标记被用作将动态数据切片，我们的实践中使用一个月的间隔将数据切为12个时间片。最后，通话的时长被用作归一化表示个体之间联系权重的依据。

我们的建模过程从原始通话记录出发利用一定的数学模型（后文详述）追条解析通话记录，把通话记录归一化为加权网络图的数据（事实表），同时提取出基站信息和个人信息作为维表待用。最后使用前文提到扩散-剪枝算法框架和临界事件判定算法对这个有权网络图进行分析，得到分析结论。表示这个挖掘过程的数据集市示意图如图4-1所示：

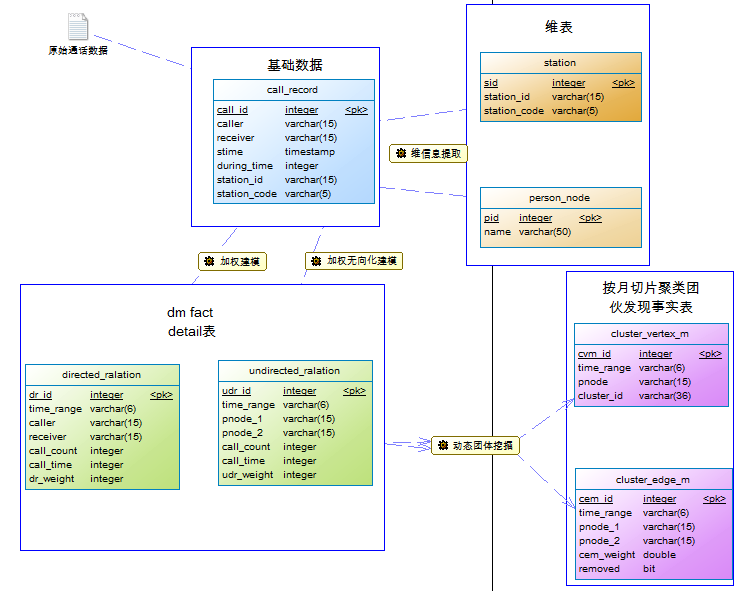


图5- 1 基于通话记录挖掘团伙并侦测临界事件的数据集市示意

建模的关键在于将通话记录归一化为节点之间的权重。我们认为，人与人之间只要发生了通话，无论通话时间长短，就具有一定的联系权重。另外，通话时间也是作为权重高低的一个衡量标准。基于这样的考虑，我们的权重计算数学模型如下：逐一考虑两个人的通信记录，对于每条记录，先记30point（权重得分），通话时间超出30秒不足60秒的部分，每秒计1point；超出60秒不足120秒的部分，每秒计1/2point；超出120秒不足180秒的部分，每秒计1/3point……最终叠加得出两两个人之间的联系权重信息。数学公式如公式4-1所示：（设a与b的权重为w(a,b)，两人的各条通话记录为Ri(a,b), i≥1，每条通话记录中的通话时间为Ri(a,b).t）

——公式4- 1

其中δ(x)是kronecker delta，即克罗内克符号函数：

通过上述建模，我们的通话记录数据集就可以转化为有权的社会关系网络图了，可以利用前文提到的一些分析算法对其进行挖掘。

实验设计与分析

本节实验使用的数据来自一份真实的通话记录数据，包含255个节点一年内产生的460,369条通话记录。但是这份数据的网络结构并不好，由于隐私权限制，我们取得的255个节点数据，是这255个节点主叫和被叫的通话记录。这些记录涉及到3000余个节点。初步的实验中还发现，这255个节点只有60多个有相互联系。所以为了更真实的反映社会网络情况，我们队这份数据做了增广模拟扩展，将所有涉及到的节点都包括到网络中来。在下面的实验中，我们使用的数据集就是这份扩展模拟的通话记录数据，共包括3,264个节点，72,354,329条通话记录。

数据集比较庞大，我们下面将采用前文提到的扩散-剪枝算法框架从一个典型节点40159533210出发扩散提取出与该节点有2度关联的数据并使用5.2提到的数学模型进行归一化处理成有权无向图，然后以一个月为时间间隔将数据分为12个时间切片，在每个时间片上使用B-M算法对这个子网进行聚类分析，提取出与关键节点有关的团伙。最后再使用临界事件侦测算法发现关键节点及相关团伙的演化过程中发生的关键事件。

由于原始数据很大，扩散和建模的过程我们在一个包含200个节点的hadoop/hive集群上使用mapreduce程序完成，主要使用java, python, hiveSql和shell编写。后续的步骤利用前文已经实现的java程序在单机上进行处理。

### 步骤1 扩散

在我们的实验中，从关键节点40159533210出发，提取出原始数据中与关键节点有2度关联的所有节点记录（即迭代3.4提出的算法中的步骤1两次）。得到的子网数据集还包含217个节点。

然后使用一个月为时间间隔将数据分成12个时间片，在每个时间片上利用5.2提到的数学模型进行归一化处理，每个时间片得到一个有权无向网络图。得到的每个时间片上包含200个节点，3300条边左右。

### 步骤2 剪枝

在每个时间片上，使用前文提到的B-M自动聚类算法进行聚类计算，剪去无关的节点。图5-2是其中一个时间片上包含关键节点40159533210的团伙结构图。（下文称关键节点40159533210为A）。为了保证画面的清晰可辨，图中去除了id的显示，并只显示重量级联系（高权重边）

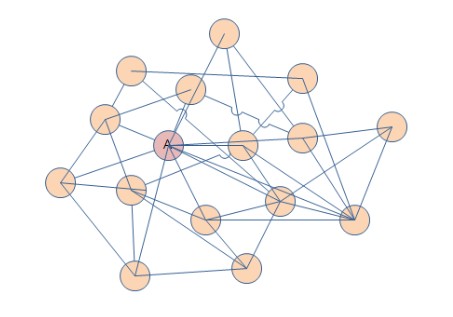


图5-1 时间切片200901上包含关键节点A的团伙结构

### 步骤3 关键事件侦测

使用第4章提到的临界事件侦测算法发现12个切片上关键节点及关键团伙的关键事件。得到如表5-1所示的结果：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 200901 | 200902 | 200903 | 200904 |
| 节点533210 | 出现 | — | — | — |
| 团d1f9607f | 生成 | k-延续(k=92)  k-成长(k=20) | k-延续(k=87) | k-延续(k=96) |
|  | 200905 | 200906 | 200907 | 200908 |
| 节点533210 | — | — | — | — |
| 团d1f9607f | k-延续(k=87)  k-成长(k=15) | k-延续(k=100) | k-延续(k=96) | k-延续(k=81)  k-衰退(k=81) |
|  | 200909 | 200910 | 200911 | 200912 |
| 节点533210 | 离开d1f9607f  进入7874c038 | — | — | — |
| 团d1f9607f | k-分裂(k=100) | k-延续(k=100) | k-延续(k=91) | k-延续(k=85) |
| 团7874c038 | — | k-延续(k=100) | k-延续(k=87) | k-延续(k=93) |

表5- 1 12个时间片上关键节点和关键团发生的临界事件

我们可以看到，我们所关注的关键节点40159533210的社会关系相对比较稳定，大多呈现延续状态。而在时间片200909上，这个团伙发生了分裂事件，成为了两个团伙，关键节点进入到其中的一个。这个活动序列可以从一个侧面反映关键节点所在的社团或社交圈的演化活动。

### 结果分析

根据上述三个步骤的实验结果来看，利用我们的算法框架对实际数据进行分析是可行的。最终得出的结果能够一定程度上反映所关注社区的演化行为。从处理时间上来看，步骤1中进行的扩散和建模过程在200节点的hadoop集群上需要共计约17分钟的计算时间。后续的步骤2和步骤3由于算法设计的限制只能在单机上运行，每个切片需要处理12分钟左右，最终需要单机处理时间超过2个小时。要应用到实际中庞大得多的社会网络，这样的处理的能力是不够的，还需要对算法进行优化和并行化改写，才能使算法效率符合实际生产的需要。

本章小结

本章针对一个实际的通话记录数据集，进行了一定的数学建模，将数据归约成有权网络图，然后利用前文提出的扩散-剪枝算法框架和临界事件侦测算法进行了分析。本章第二节提出的数据流程可以适用于在其他场景下对通话记录数据进行归一化分析，对其他的如Email通信记录、博客引用记录等也有一定参考价值。同时提出的权重建模公式尽可能地还原了联系的真实情况，对于不同的场景可以根据需要进行调整。本章还令40159533210节点作为关键节点，使用扩散-剪枝算法框架在12个时间切片上挖掘出了与40159533210节点所在团伙，进而对40159533210节点所在团伙的演化临界事件和40159533210的节点临界事件进行了侦测。定量地描述了与关键节点密切的社会网络的演化规律，证明了在实际生产中应用动态挖掘相关算法对有关数据集进行分析并提取有用信息的方法是可行的，为类似应用的实现提供了有力的理论依据。不过，这一系列算法的计算开销依然很大，还需要根据需要进一步优化算法提高效率。另外，计算的准确性也需要接受更多实践的考验。

总结与展望

本文立足于一个基于实际通话记录的团伙发现和动态临界事件侦测研究，实践和分析了其中涉及的聚类技术，并利用modularity度量设计了一种B-M自动聚类的算法。该算法能够无监视地完成团伙发现任务。在一定干扰因子的范围内，算法取得的效果良好。但该算法有较高的算法复杂度，在处理大数据时有较大的局限性。考虑到实际应用中网络庞大、关心的区域较小的情况，本文又提出了扩散-剪枝的算法框架，只提取出网络中我们所关心的区域进行计算，从应用层面降低了计算的消耗。

在完成单个时间片上的团伙发现后，本文还使用侦测临界事件的方法对多个时间片上团伙的演化行为作了一定的分析和研究。我们发现，定量定义下的关键事件侦测的准确度主要依赖于聚类的准确度，如4.3所述。另一方面，临界事件的定义还停留在比较理论的阶段，还需要更多的实践来证实和改进。

本文最终使用提到的团伙发现算法和临界事件侦测算法对一个真实的通话记录数据集进行了分析，挖掘出其中的关键节点和关键团伙及其演化过程中发生的临界事件。证明了将这一系列算法应用到实际应用中的可行性，为相关应用提供了有力的理论依据。

但本文中提及的算法还有很多局限性。其一就是团伙发现算法（B-M聚类算法）巨大的计算消耗和串行化特性。作为未来的工作，有必要进一步优化算法效率并研究并行化的算法，才能在实际应用中有较好的可用性。其二就是文中提到的临界事件，还停留在理论研究的阶段，未来的工作中需要更多的实践来优化和完善临界事件的定义和侦测算法，使我们的研究能真正应用到实际的生产实践中去。

参考文献

1. M.E.J. Newman, M. Girvan, Finding and evaluating community structure in networks, Phys. Rev. E 69 (2) , (2004) 026113.
2. M.E.J. Newman, From the cover: Modularity and community structure in networks, PNAS 103, (2006) 8577-8582
3. S. Asur, S. Parthasarathy, D. Ucar, An event-based framework for characterizing the evolutionary behavior of interaction graphs. In Proc. of KDD'07, (2007) 913-921
4. G. Palla, A.-L. Barabási, T. Vicsek, Quantifying social group evolution, Nature 446, (2007) 664-667
5. Nina Mishra, Robert Schreiber, Isabelle Stanton, and Robert E. Tarjan Clustering Social Networks, Computer Science, 4863 (2007), 56-67
6. Chayant Tantipathananandh, Tanya Berger-Wolf, David Kempe, A Framework For Community Identification in Dynamic Social Networks. In Proc. of KDD'07, (2007) 717-726
7. Tanya Y. Berger-Wolf, Jared Saia, A Framework for Analysis of Dynamic Social Networkss. In Proc. of KDD'06, (2006) 523 – 528
8. Ulrik. Brandes, A faster algorithm for betweenness centrality, J. Math. Sociol. 25 , (2001) 163-177
9. Santo Fortunato. Community detection in graphs. Physics Reports 486, (2010) 75-174
10. H. W. Kuhn. The Hungarian method for the assignment problem. Naval Research Logistics Quarterly 2, (1955) 83–97
11. Marques JP, Written; Wu YF, Trans. Pattern Recognition Concepts, Methods and Applications. 2nd ed., Beijing: Tsinghua University Press, 2002. 51−74 (in Chinese).
12. Brandes, U., Delling, D., Gaertler, M., Görke, R., Höfer, M., Nikoloski, Z., Wagner, D.: On Modularity Clustering. IEEE Trans. Knowl. Data Eng. 20, (2008), 172-188
13. Y.-R. Lin, Y. Chi, S. Zhu, H. Sundaram, B.L. Tseng, Facetnet: a framework for analyzing communities and their evolutions in dynamic networks, in WWW'08: Proceedings of the 17th International Conference on the World Wide Web, ACM, New York, NY, USA, (2008) 685-694.
14. A. Strehl and J. Ghosh, Cluster Ensembles— A Knowledge Reuse Framework for Combining Multiple Partitions, J. Machine Learning Research, vol. 3, Dec. (2002). 583-617,

致谢

值此论文完成之际，回顾自己的学习科研生活。整个研究过程中，我承受了相当多的关爱、鼓励和帮助。借此机会，我向所有关心和帮助我的人表示由衷的感激之情。

饮其流时思其源，成吾学时念吾师，在此论文完成之际，谨向我尊敬的导师何钦铭教授致以诚挚的谢意和崇高的敬意。何老师渊博的专业知识、务实严谨的科研作风、宽以待人的处世态度使我受益匪浅。本文的最终得以成文，离不开您一路上对我的指导和启示。

还要感谢指导我进行研究的学长——马连航博士。从确定研究方向，查阅文献到算法的设计与实现，再到实验的设计和结论的总结，直到最终论文的完成，都饱含学长对我的指导和点拨。

感谢浙江大学，在浙大的四年本科学习，我不仅学会了丰富的计算机知识，更重要得是学会了科学的学习方法和研究方法。这四年来的收获将是我人生中最宝贵的财富，令我受益终生。

感谢我的家人和朋友，我的学业的顺利完成同他们的支持鼓励和理解是分不开的，感谢他们的帮助和宽容。

最后，还要感谢所有曾经帮助过我的，由于篇幅限制未被提及的人们，感谢你们对我的关心和帮助，谢谢！

**本科生毕业论文（设计）任务书**

**一、题目：**

**二、指导教师对毕业论文（设计）的进度安排及任务要求：**

**起讫日期 200 年 月 日 至 200 年 月 日**

**指导教师**（**签名） 职称**

**三、系或研究所审核意见:**

**负责人**（**签名）**

**年 月 日**

**毕 业 论 文（设计） 考 核**

**一、指导教师对毕业论文（设计）的评语：**

**指导教师(签名）**

**年 月 日**

**二、答辩小组对毕业论文（设计）的答辩评语及总评成绩：**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **成绩比例** | **文献综述**  **占（10%）** | **开题报告**  **占（20%）** | **外文翻译**  **占（10%）** | **毕业论文（设计）质量及答辩**  **占（60%）** | **总评成绩** |
| **分**  **值** |  |  |  |  |  |

**答辩小组负责人（签名）**

**年 月 日**