# 摘 要

由于实时复杂网络快速、连续到达的大量数据使大多数的传统的聚类方法失效，因此基于数据流的复杂网络社团发现是一个难题。 传统的数据流社团发现算法不能满足应用要求并制约其发展。所以，研究并设计实时复杂网络的社团发现算法，意义重大。

本文研究了传统的社团发现和数据流聚类方法，分析了各种算法的优缺点，在传统的实时社团发现方法、数据流核心技术和经典的数据流聚类算法的基础上，设计了一种新的复杂网络的社团发现算法。该算法首先研究了网络中结点之间相似度量方法，提出了将基于复杂网络的社团发现转换成一个N维空间的聚类算法。借鉴D-Stream算法的框架，将复杂网络的社团发现算法分解为在线处理层和离线聚类层；在线处理层作为缓冲区快速读取数据流中到达的数据，将每一个新格结点映射到对应的网格单元和区域，形成相关的记录，并将其存储在基于网格的特征向量中。利用权重随时间衰减的方法，来满足对历史数据流信息聚类的演化分析需求；离线层对信息进行精确聚类，用网格的特征向量的密度作为聚类的依据。利用这两层处理框架实现了数据流聚类算法的精度和速度的折中。提高聚类的质量，网格单元能细分之后构成的网格作为聚类基本单位，如果网格单元密集并且强连接，就将聚类合并，为此本文提出了网格相邻判断以及合并算法。利用树状数据结构解决了一些不相交的聚类的合并和查询问题，更有效地支持了复杂网络的社团发现算法的实现[1]。

通过实验，该算法可以发现任意数目、任意形状的社团，还能有效去除噪声，是一个具有应用前景的适用于数据流复杂网络的社团发现算法。

**关键词：数据流 复杂网络 社团发现 网格与密度 聚类**

# ABSTRACT

With the real-time complex network fast, continuous arrival of large amounts of data, most of the traditional clustering method fails. Traditional stream-based community discovery algorithm can not meet the application requirements and restricts its development. Therefore, the research and design complex real-time network community discovery algorithm is significant.

This paper has studied the traditional community discovery and data stream clustering method and analyzed the advantages and disadvantages of various algorithms. Based on traditional real-time community-based approach, data flow core technology and classic data stream clustering algorithm, we designed a new complex network of community discovery algorithm. Firstly, the study of the similarity measure between the nodes in the network method, proposed a complex network of community-based discovery into an N-dimensional space of clustering algorithms. refer to the method of D-Stream algorithms, complex network of community discovery algorithm is divided into online processing layer and offline clustering layer; online processing layer as a buffer quickly read incoming data stream, forming the relevant records, and will feature vectors of the grid based on their storage. With weight of data decay, the information is accurate off layer clustering feature vector with the density of the mesh as the basis for clustering. Use this framework, we improved the quality of clustering, if the grid cell density and strong connection, it will be merged. Using tree data structure to solve some consolidation and queries regarding disjoint clusters had effectively support algorithm implementation.

Through the experiment, the algorithm can be found in any number, any shape society, but also effectively remove noise, is a promising community for data stream complex network discovery algorithms.

**Key words: data stream complex network community grids and density**

# 目 录

第一章 绪论 1

1.1社团发现的选题背景及意义 1

1.2国内外研究现状 2

1.3论文主要工作及创新点 3

1.4论文组织结构 4

第二章 复杂网络的社团发现算法综述和分析 5

2.1复杂网络与社团检测 5

2.1.1 网络的定义 5

2.1.2基于全局比较的社团定义 6

2.1.3基于局部指标的社团的定义 6

2.2基于网格与密度的数据流聚类的定义及特点 7

2.2.1基础定义 8

2.2.2引力 9

2.2.3基于密度的网格簇 12

2.3 主流的社团发现算法的分类 14

第三章 基于网络与密度的实时社团发现聚类算法 17

3.1 TNV算法（Transform Node to Vector） 17

3.1.1基本概念 17

3.1.2 TNV算法 18

3.2基于网格和密度聚类算法 19

3.2.1 网格检测以及时间间隔gap 19

3.2.2 检测以及移除稀疏网格 20

3.2.3相邻网格的合并 24

第四章 实验及分析 29

4.1实验环境和测试数据 29

4.2社团发现的演化过程与效果分析 29

4.3社团发现算法性能比较 30

4.4时间和空间复杂度性能比较 31

致 谢 35

参考文献 37

# 第一章 绪论

1.1社团发现的选题背景及意义

近年来，由于传感器技术和计算机技术的广泛应用和互联网的普及，数据的获取越来越容易和趋于自动化。在网络控制、通讯行业、自动化、股票市场、气象环境监控、web等很多领域产生了新的基于复杂网络关系的数据流。复杂网络一般指结点很多、联系关系复杂网络，因为它的灵活普适描述能力，可以广泛地应用于各个科学领域，从而对复杂系统进行分析建模，近年来吸引了更多的人对它研究。随着深入研究，许多现实网络都有社团结构，即复杂网络由很多个社团构成，社团之间的连接边比较稀疏、而社团内部的连接边相对稠密[1]。

社团是网络中结点所组成的分组，组内的边比较多，而组间的边比较少。复杂网络社团结构有很重要的意义，因为社团结构经常被应于网络中的某功能单元，例如，互联网中有相似或相同主题的站点组成的社团；生物网络中有相似功能的组织。发现复杂网络中社团可以看作是对复杂网络进行聚类过程。社团发现也可以通过规律揭示某个具体的节点在复杂网络起的作用。

在数据挖掘中，分析可以对数据对象的未知类进行标识，同时通过找最大相似性和最小相似性来分组和聚类。而流数据挖掘与传统的数据集不同，这些数据是海量（massive）的、时序的（temporally ordered）、快速变化和潜在无限的（potentially infinite）。海量数据导致不可能在内存或硬盘存储整个流数据。不仅太多的数据，而且要记录的属性值相当多。同样式因为数据量的巨大，多遍扫描数据并挖掘的方法是不切实际的，因此数据流挖掘应是单边扫描过程。数据流快速变化，因此看到数据流中每个数据元素是不可能的，只能分析部分的数据元素做出决策。因为数据流的时序性，所以对流数据元素访问是单次线性。那么数据元素只可以按它流入顺序依次读取一次，而随机访问不现实[1]。多数应用需要很快响应时间，挖掘应该是个连续、在线过程，而不应是偶然地进行一次；数据流往往天生就是高维的。聚类是一个发现密集和稀疏的区域以及各区域之间相互联系的过程，从而发现数据的全局分布模式,以及数据属性之间有趣的关系。数据流聚类是传统聚类方法在大数据数据流环境下的延伸, 是一种重要的并具有强大信息提取功能的数据约减技术,有着广泛的应用前景。例如,在新兴领域它可以应用于传感数据流聚类,气象卫星遥感、网络通信量监测和电力供应等等;可以对网络使用情况进行监控,从而掌握网络用户的使用方式并对异常的网络使用行为进行判别;它还可以应用在诸如电信记录的管理与分析,商业交易管理与分析及金融信息监控与分析等众多的应用场景之中。

社团发现作为一种特殊的流聚类，一直是一个热点研究问题，然而现有方法还远远不能满足时代的需求。本文现有方法大多只能处理静态不变网络数据，而且由于网络这种结构的复杂性，现有算法往往不能很好解决大规模数据量的计算问题。针对这些问题，本文分别从理论算法和实现应用两个角度出发，设计了一种新的复杂网络社团发现算法。

1.2国内外研究现状

在实时的数据的聚类研究问题上，早在1987年提出一种名为CobWeb的方法。该方法将数据用分类树进行组织，通过衡量聚类分类效用启发式的进行结点合并、分裂、插入等操作。另一种经典的算法BIRCH利用聚类特征概括簇描述，克服了静态聚类算法可伸缩性差等缺点[4]。

另外流数据聚类研究也引发了研究工作者的广泛关注。 K-means是该领域最早提出算法之一，该算法处理数据流过程中把数据点聚类为k个簇，使得每个数据点和其簇中心误差平方和最小。韩家炜等研究工作者在2003年提出了演化聚类的框架Clustream，创造性地把聚类分成在线与离线两个部分[1]。

国内的傲英、曹锋、钱卫宁等学者在2006年提出了一种DenStream算法，将聚类过程分成在线划分微簇、离线处理用户聚类请求两部分。南京航空航天大学的屠莉等在2007年提出一种局域密度的网络的实时数据流聚类算法D-Stream。

国外在数据流挖掘方面有两个较有影响力的研究小组。一是斯坦福大学R.Motwani教授的研究小组,二是伊利诺伊大学厄巴纳-香槟分校的C.Aggarwal和J.Han教授领导的研究小组。R.Motwani教授的研究小组研究方向侧重在数据流管理、数据流的聚类和数据流的连续查询方面。C.Aggarwal教授领导的研究小组具体研究方向更侧重在数据流分析方面,至于数据流在线分析,在分类、聚类 、频繁项集挖掘、以及可视化等方面做了大量的研究工作,提出了倾斜时间策略,采用不同时间粒度保存数据流的信息[1]。

随着聚类算法数据的处理效率的要求及对聚类质量不断提高，以及聚类算法在互联网技术中的应用越来越广泛。Clique算法是基于网格的聚类算法中的一个经典算法,它是Rakesh Agrawal等人最早提出的。此算法将数据每个属性进行等分,把数据空间划分为网格结构,然后该算法根据每个网格单元中数据点数量来发现其中的稠密单元,然后对稠密单元进行合并就可形成聚类。该算法能处理大量高维数据,并可以自动识别嵌入在数据子空间中的聚类。另外, 基于网格的聚类算法还有Sting算法、WaveClustetr算法以及它们的一些改进算法[1]。

2006年,曹锋等人提出了DenStream算法。相对于CluStream算法有较大的改进,继承了Incremental DBSCAN基于密度的优点, 支持对有噪声的动态进化的数据流进行任意形状的聚类。同年，中山大学的朱蔚恒等在文献中提出的基于密度与空间的AClustrearn聚类算法,朱蔚恒等人针对CluStream算法不适用于发现任意形状的聚类以及对周期性数据的聚类变化反映不完整的问题,提出一种采用空间分割、组合以及按密度进行聚类的算法ACluStream[1]。

尽管已经存在大量的算法和处理框架，但是由于数据流的特点，基于复杂网络的社团发现算法数据流聚类分析还面临着许多挑战，例如动态适应性低，不能处理混合属性的数据，这也是很多传统聚类算法所存在的问题。实时的社团发现算法仍然是一个需要继续深入研究的课题。

1.3论文主要工作及创新点

本文根据复杂网络的社团发现的特点,分析动态实时复杂网络社团发现的要求以及在流聚类方面的最新研究成果。在对目前复杂网络的社团发现算法与传统算法进行比较分析的基础上,给出基于密度和网格的数据流聚类算法。 算法能够有效分离噪声数据,发现任意形状的聚类。该算法应符合数据流特点和数据流算法要求,具有较高的处理速度和较高准确性。本文的主要研究工作和创新点包括以下几个方面:

（1）深入研究了复杂网络社团发现问题背景意义和当前已有算法优缺点。利用聚类发现复杂网络社团结构，并指出了具体流程。

（2）分析了一种迭代算法，将复杂网络中的结点映射为N维空间中的向量，向量空间可以利用相似度量方法确定网络中结点的相似性。

（3）使用基于网格与密度的算法实现复杂网络社团发现，并且使用一种网格自适应的划分方法。为了满足系统实时性，提出了衰减函数，并将网络作为聚类的最小单位，提出了密集网格区域相邻判断和合并方法。

（4）适用了一种最小生成树的算法,对网格进行快速合并。

1.4论文组织结构

第一章绪论。阐述本文的研究背景、意义，对本文研究领域进行简要介绍，阐述国内外的研究现状，并介绍本文所研究的主要内容。

第二章复杂网络的社团发现算法综述和分析。首先给出了复杂网络与社团发现的一些基础知识，包括网络及社团的定义、社团结构的度量标准；最后网络中与数据流聚类分析紧密相关的技术及概念进行介绍。并比较不同算法的适用条件和优缺点。结合现有数据流聚类算法分析数据流处理中的关键技术点。最后，总结各类社团发现算法的优缺点。

第三章 基于网格与密度的数据流聚类算法。本章在提出基于网格和密度方法和数据流聚类算法，首先提出新的衡量网格密度的方法，对时间框架进行改进，对存储结构进行了改进，然后设计并实现一个完整地基于网络与密度的数据流聚类算法GDCD。

第四章 实验及分析。应用人工数据集和真实数据集对本文研究的社团发现算法的性能进行验证，并对实验结果进行分析比较。

# 第二章 复杂网络的社团发现算法综述和分析

复杂网络在现实生活中有广泛的应用，例如在社交网络、蛋白质网络、互联网等领域，利用这种数学模型，可以进行许多特定问题的研究，例如需找网络中最有影响力的节点、找出经常出现的特定结构、进行分类发现社团等。本章将首先阐述社团发现相关概念，包括网络与社团的定义、社团结构的评价指标，然后对一些经典的社团发现算法进行了概述，列举了一些经典的一些社团发现分析算法。

2.1复杂网络与社团检测

2.1.1 网络的定义

**定义2.1 网络**(Network）网络是用来生活中网状事物进行数学建模的数据结构，可以用二元组N=(V,E)来表示。（V为结点集合，表示生活中的单个物体，E为个体间联系的集合，称为结点的边）。如果网络中边E的连接关系是不分方向，称为无向网络，反则有向网络；如果网络中的边上带有某种特定意义的数值，那么称它为加权网络，反之无权网络。

从开创图论研究的早期，人们对于复杂网络这种数据结构的认识十分有限。近年来，由于互联网和物连网等复杂网络的出现，复杂网络的研究得到了重视。并出现了复杂网络的无标度性和小世界性两种说法：前者指出真实网络中结点的度的数目呈幂律分布，少量的节点连接着大多数的其他结点；后者提出真实网络中存在着大量的短路径，例如著名的六度理论：即任意两个结点的距离很大概率小于等于六。这些性质表明真实网络远比人们预想的复杂，它使随机与有序的折中。此外，复杂网络有其他特殊性质，如复杂网络中是存在着大量的社团结构得。

社团结构的存在理论上与复杂网络的属性紧密相关。在现实生活中，也很容易社团存在的必然性。例如生物网络，密切联系的细胞形成了特定功能的组织；在社交网络中，人们可以因为一些共同感兴趣的爱好而形成圈子。组织和圈子都是现实生活中的社团结构。但是因为复杂网络多样性，迄今尚无完全通用的社团的定义，主流的定义主要两种形式：基于全局比较的社团定义和基于局部指标的社团定义。

2.1.2基于全局比较的社团定义

基于全局比较的社团定义要求社团内结点的关联较紧密，而社团外结点的关联较松散。社团的严格程度不同，定义两种：强社团、弱社团[4]。

**定义2.2 强社团**(Strong Community)网络N=（V，E）的子集网络C为一个强社团，当且仅当式（2-1）

 式(2-1)

其中表示节点i在C入度，代表节点i与子集C的出度，该定义表示强社团的所有结点都满足与社团内部的边的数量大于同社团外部的边的数量。

**定义2.3** **弱社团**（Weak Community）网络N=（V,E）的节点子集C为弱社团，当且仅当式（2-2）所示

 式(2-2)

其中表示社团C内部入度之和，表示社团C中出度之和。即弱社团内部节点的边数量之和大于社团外部的连边数量和。

2.1.3基于局部指标的社团的定义

除此之外，另一种较常见思想是先检测出社团的某局部结构，例如团、核等，然后用连通性把这些局部小结构连接成社团[4]。

**定义2.4 团**（Clique）网络N = (V,E)的子集C为团，如式（2-3）

 式(2-3)

dist(i,j)表示团内节点i和j的距离，即该定义要求团中任意两个节点的距离不超过k。

**定义2.5** **核**（Core）网络N=(V,E)的结点v为一个核，当且仅当该节点满足

 式(2-4)

其中adj(v)为v节点的邻居点集，表示邻居个数，即该定义的意思是邻居数大于等于预先指定的阈值k。

这些定义识别了网络中存在的微小局部结构，通过连通性能发现较大的社团，而这些方法充分利用了基于网格和密度聚类的某些思想。

2.2基于网格与密度的数据流聚类的定义及特点

实时聚类高维数据流是社团发现算法的一个难点。因为现实生活中，在大数据的环境下，数据越来越实时，在一些应用中，大体积的多维数据流到达实时的数据收集中心。原始数据通常拥有庞大的体积， 可以仅扫描依次后的时间顺序。重新如何存储，查询和分析数据，出现了积极地研究流。聚类是一项关键的复杂网络的数据挖掘任务。而挖掘数据流有有几项关键的挑战：首先，单遍扫描；第二，将数据流视为数据一个很长的向量在很多应用中捉襟见肘，用户更加关注簇的演化行为[8]。

近来，出现了许多数据流聚类方法。比如D-Stream、CluStream以及扩展（在多数据流，分布式数据流，并行数据流上的扩展）等。

CluStream以及扩展的算法有以下一些缺陷：

（1）只能发现球形簇，不能发现任意形状的簇。

1. 不能够识别噪声和离群点。

（3）基于k-means的算法需要多次扫描数据（其实CluStream利用两阶段方法和微簇解决了该问题）。

基于网格与密度的聚类算法介绍。基于网格与密度的方法可以发现任意形状的簇，可以处理噪声，对原始数据集只需一次扫描。而且，它不需要像k-means算法那样预先设定k值。

文本提出了GDCD,一种基于密度的数据流聚类框架。它不是简单用基于密度的算法替代k-means的数据流算法。它有两项主要的技术挑战**：**

首先，本文不将数据流视为静态数据，本文对数据流的演化时间特征更感兴趣。为了捕获和分析簇的动态过程，本文研究了一些想法比较新颖，这种方法可以将衰减因子与每个格子的密度联系起来。与CluStream要求输入聚类目标的所持续的时间不同，衰减因子能通过把最近的数据赋更高的权重并不会完全丢弃历史的信息，从而动态地、实时自动地形成簇。另外，GDCD不用输入簇的数目。因此，GDCD特别适合于那些对应用程序数据具有少量领域知识的用户。

其次，由于数据流的数据是海量的，不大可能保留每个数据记录的密度信息。因此本文提出将数据空间划分成一个个离散的网格，然后将新来的数据映射到对应的网格。这样，本文不需要保留原始数据，仅仅需要操纵这些网格。然而，对于高维数据，这些网格的数目是海量的。因此如何处理高维数据来提高伸缩性是一个关键的问题。幸运的是，实际上大部分网格是空的或者只包含少量的记录，本文在GDCD中开发了一种内存有效的技术来管理这些松散的网格。

与CluStream算法比起来，GDCD在聚类质量和效率方面更有优势，而且针对海量高维的流数据具有更好的伸缩性。

2.2.1基础定义

**定义2.6向量空间**(Vector Space) 假设输入的数据有d。并且在以下数据空间中定义：

 式（2-5）

**定义2.7密度网络**（Density Network）将d维空间S划分成密度网络。假设对于每一维，它的空间是被分成个部分。

 式（2-6）

这样数据空间S被分成了个密度网格。每个密度网格g是由组成，本文将它表示为：

 式(2-7)

一个数据记录可以映射到下面一个密度网络g(x)：

 式(2-8)

**定义2.8特征向量**(Feature Vector)一个网格g的特征向量是一个五元组：是g更新的最后时间，是g作为松散（稀疏）网格从网格集合中移除的最后时间，D是网格最后更新后的密度，label是网格的类（簇）标签，status={稀疏,正常}是一个用来移除稀疏网格的标签。

**定义2.9密度系数**(Density Coefficient)对每个记录x，本文分给它一个密度系数，它随着时间递减。实际上，如果x在tc时刻到达，本文将它的时间戳定义成：T(x)=tc,这样它在时刻t的密度系数D（x，t）就是：

 式(2-9)

任何网络的密度是经常变动。然而，本文发现没有必要在每一个时刻去更新所有数据记录的密度值，而是，只有当一个数据点映射到那个网络时，本文去更新网络的密度。对于每个网格，当一个新的数据记录到达某个网格，且接收需要记录最后的数据记录是，本文才根据下面的结论去跟新网格的密度。

2.2.2引力

对于一个方格g，让它的在第i维的长度为并让为第i维中间点的位置。对于每一个数据记录映射到g，构造一个中心为x长度为,Cube(x)超立方体。令为Cube（x）和方格h的交容量，x和h的初始引力被定义为占Cube（x）容量的比率。即，

 式(2-10)

本文列举在图2-1(a)中的初始引力。正如图2-1(a)显示，数据映射到以格子g上。Cube（x）是矩形ACEG，大小为。Cube（x）和格子g，h，f和k相交。

从图2-1（a），本文能看出g和Cube（x）相交的面积。

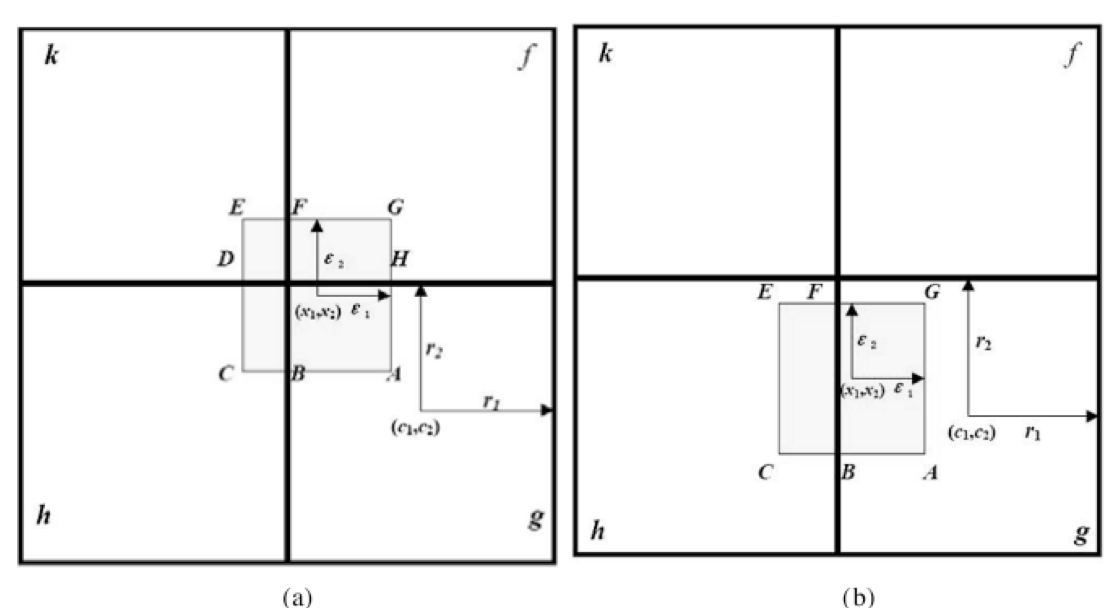


图 2-1 一个数据点和一些格子初始引力的实例

因此，x和g初始引力为



相似的，对于方格h，本文能发现





Cube（x）将会完全在i维有方格g。在这种情况下，如果h和g在第i维和

例如，在图2.1（b）中的数据x，本文有下面的结果：

下面的结果概括了d维的情况。在一个d维的空间中，一个格子g以。本文定义g的邻居，NB（g）为一系列的格子中心和g在最后一维上有差异。

**定理2.10**.对于一个d维的数据记录,让x映射到一个中心为方格g。让作为一个g的一个以为g’中心的邻格。x和引力是：

 式(2-11)

代表x和h在第i维的吸引力，即

 式(2-12)

其中;如果且如果i = k，并且。

本文有关于总引力在周围格子数据中下面的结果。

定理：对于一个d维数据记录，让x映射到一个格子g，然后本文有

 式(2-13)

证明：该定理能够从引力和邻居的定义证明出来。因为x映射到g并且超体Cube（x）有宽，，在第i维，我让Cube（x）仅仅和NB（g）的格子相交，并且交体积等于Cube（x）的面积。因此，让V（x,h）作为Cube（x）和格子h的交体积，本文有。所以得到式（2-14）。

 式(2-14)

2.2.3基于密度的网格簇

**定义2.11** **网格簇**（Grid Cluster）如果G内部每一个网格都是稠密网格而每个外部网格或者是一个稠密网格或者是一个过渡网格，那么G就是一个网格簇。

直观地，一个网格簇就是一个连接的网格组，它比它周围的网格密度要大。注意到本文总是尝试在任何可能的时候合并簇，因此这样会导致簇被松散的网格所包围。

**定义2.12格子组**（Grid Group）一群格子是一个格子聚类如果他是一个格子组，每个内部的格子G是一个密格子，每个外部的格子G既不是一个密格子也不是一个过渡格子。

直观地，一个网格簇是具有更高的密度比周围的栅极连接的栅基。需要注意的是，本文总是试图尽可能合并簇，因此所产生的集群由稀疏的网格包围。

**定义2.13邻格**（neighborhood grids）对于两个密格子且如果存在k。,并且

1. 并且

（2）,

和在第k维度是邻接格子。

**命题2.14** 让从时刻0到时刻t到达的所有数据记录成为一个集合。本文有：

 式(2-15)

**命题2.15** 表明系统中所有数据记录密度的总和永远不会超过。由于存在个网格，因此每个网格的**平均密度**不会超过。有这个观察得到以下的定义：

在时刻t，对一个网格g，如果有

 式(2-16)

本文称它为稠密网格。其中，>1是一个控制阈值的参数。比如，本文设定=3。本文要求N>，因为D(g,t)不能超过。

在时刻t，对一个网格g，如果有

 式(2-17)

本文称它为稀疏网格。其中，0<<1。比如，本文设定=0.8。

在时刻t，对一个网格g，如果有

 式(2-17)

本文称它为过渡网格。

在多维空间，为了形成簇，本文考虑连接各个邻接的网格，本文按以下定义：

**定义2.16**（网格的特征向量）。本文定义一个网格单元的特性矢量作为一个元组（，，C，D，label），其中是当克被更新的最后时间，t m是当g的从网格列表中移除作为零星最后一次格（如果有的话），C是一个二维向量表示从吸引力它的二维邻居，D是在上一次更新的网格密度和标签是网格的类的标签。

本文现在需要决定如何推导基于所述密度信息簇。本文的方法是基于以下的观察。

**定理2.17**让X（t）成为所有数据记录集合从0到t，有：

1.  式（2-18）
2. . 式（2-19）

证明。对于给定的时间t，是逐一分别到来0,1,...,t的t+1数据记录密度系数的总和。对一个在时间数据记录x，，他的密度是

数据X到达时间，，其密度为。因此，该总和的所有数据记录是;

** 式**(2-20)

同样，这也是明确的：

** 式**(2-21)

**定理2.18**表现在系统中的所有数据记录的密度和将不会超过**。**

在时间t，对于一个格子g来说，本文将它称为一个稀疏格子如果  式（2-22）

其中.例如，本文设定。

在时间t，对一个格子g来说，本文将它称为一个过渡格子如果

**** 式 (2-23)

2.3 主流的社团发现算法的分类

（1）**社团亲近度算法**：算法基本原理是：判断一个人是否属于某个社团，看这个人和社团内部关系是否比社团外部关系多，如果和社团内部关系比较多，就很可能是属于这个社团。

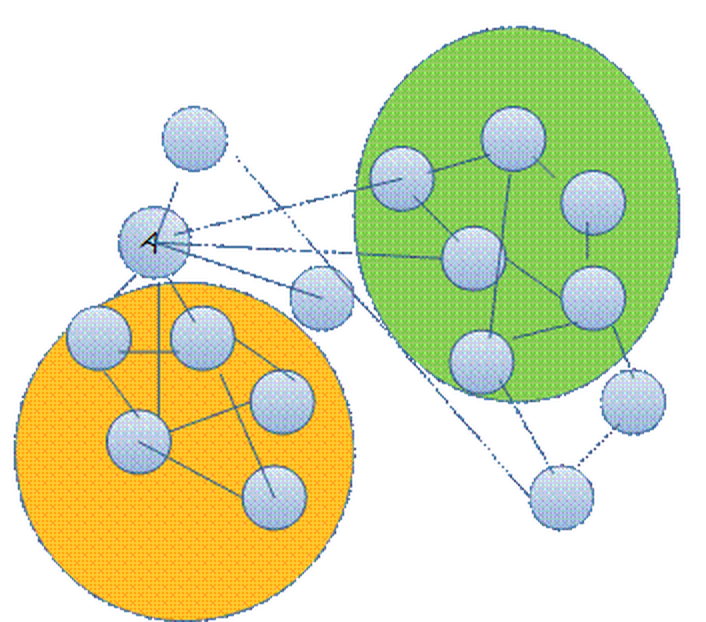


图2-2 社团亲近算法举例

假如上图2-2使我们SNS网络中部分节点，本文找出两个部分社团，分别用黄色和绿色圈住的部分。现在我们要继续计算，判断A是否也属于这两个社团中的一个。我们计算A的边。A节点的度是7，和黄色社团度=3，和绿色社团度为2。我们使用一个目标函数来量化社团。。我们使用这个公式表示社团凝聚度，意思就是Q值越大，越有可能是一个社团。

上图我们尝试把A分别假如黄色和绿色社团，计算Q值大小，如果Q值增加，并且增加最多的话。那么A应该属于哪个社团一成员，应该加入该社团。

（2）**社团密度算法**：这是利用几何学原理来计算。前面算法计算出来呃社团可能会存在“杂质”，所谓“杂质”，就是在某人同学社团里面可能混入一个不是同学的成员。用这个密度算法可以很好的“去噪”。所以两种算法结合起来。可以得出很精确地结果。

算法原理是认为，一个社团内部成员必定相互联系多，这个算法是计算联系度的。相信每个成员是多边形的每个顶点。一个三角形最多有3条边，四边形最多有6条边，五变形最多有10条边。N变形最多N(N-1)/2条边，如图2-3。

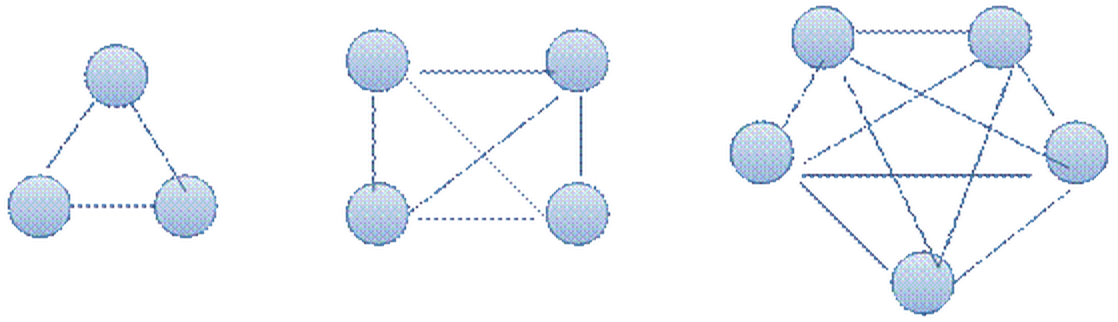


图2-3 社团密度算法举例

我们假设最大密度是1，那么密度m =实际边数\*2/N(N-1);m越大，说明越有可能是一个社团。具体算法如下：

将A假如该社团，计算其m值，如果m增大，说明应该是该社团成员。或者m值可以控制在0.7大小。大于这个值，应该是属于社团成员。

（3）**迷宫算法**：这个算法是利用游戏迷宫原理。同样，一个社团内部必定相互联系很多，把这些联系看着迷宫的一条条路。这样一个社团必定组成一个复杂的迷宫。图2-4。

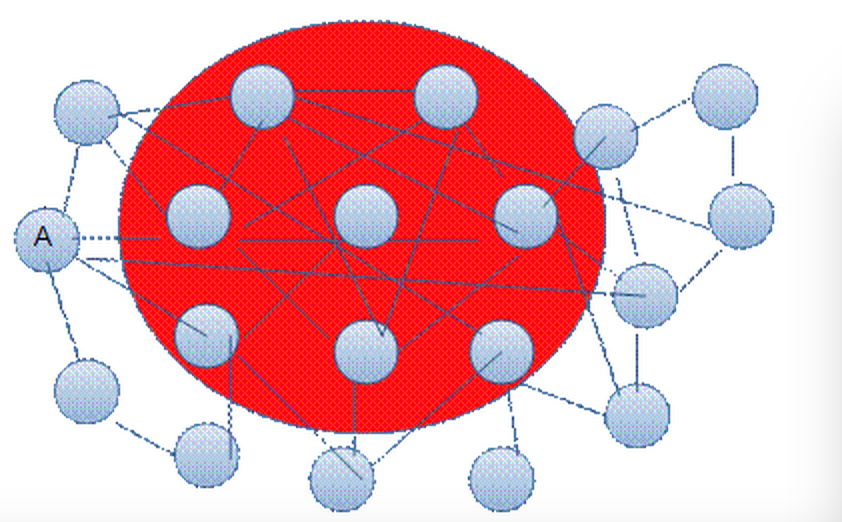


图2-4 迷宫算法举例

现在判断A是否属于红色社团，就假设一个游戏者，从A点出发、完全随机沿着边走，如果走10次还在红色区域类，说明A应该属于该红色社团成员。当然这个次数10可以自己控制。

（4）聚类复杂网络的另一个观点是:将复杂网络转化为向量的聚类.通过每个网络节点映射一个合理的N-维向量，该算法能把基于网络聚类问题转化为传统的空间点聚类问题,然后采用K-means以及基于网格的经典聚类算法对这些空间点聚类。事实上，这个思想早在针对图分割问题所提出加权二次型变换中就见雏形。提出结合谱方法并和基于向量点聚类方法的复杂网络聚类算法。首先通过计算拉普拉斯矩阵的N个最小特征向量将网络映射到N-维空间中,然后采用基于网格和密度的聚类算法。

表2-1 社团发现算法总结

|  |  |  |
| --- | --- | --- |
| **算法** | **优点** | **不足** |
| 社团亲近度算法 | 在知道入度出度的情况下，可以快速判断某个结点集合是否是一个社团。 | 只能用于社团判断，如果用于社团发现，复杂度太高， |
| 社团密度算法 | 比较简洁得计算并判断社团。 | 需要对每个集合进行判定，复杂度太高，并且不满足数据流的实时性。 |
| 迷宫算法 | 以每个点为单位进行社团发现，减少了复杂度。 | 实时性不高，数据流的情况下不好。 |
| 向量聚类 | 比较好解决了数据流的问题。 | 对大数据环境下还存在挑战。 |

# 第三章 基于网络与密度的实时社团发现聚类算法

本节提出一种新的社团发现聚类算法(GDCD)，先将社团的复杂网络结构转换成向量，之后在将一系列N维空间中的点向量映射到网格中，以网格为基本单位进行聚类。

算法框架如下：

第一步、初始化参数；

第二步、用结点转换向量算法(TNV算法)得到数据点record\_x = ;并投影到相关格子中。

第三步、对格子进行状态(稀疏，过渡，密集)更新，并将待聚类的格子存储到树状存储结构中;

第四步、每隔一段时间，对格子进行合并和查找所属聚类;

第五步、数据流缓存区是否还有数据，是则继续 第二步 。

本章先介绍将节点转换成空间中向量的方法，为了解决社团发现问题，后介绍基于网格与密度的流聚类算法。

3.1 TNV算法（Transform Node to Vector）

3.1.1基本概念

本节主要提出把网络中的结点转换为向量的算法,然后在这个空间中用欧式距离,夹角余弦等测量方法衡量结点之间的相似程度。如何将复杂网络的结点映射为向量空间的一个向量。首先,通过结点相互的传递信息得到网络中的所有结点对整个复杂网络的影响,下一节将会看到这个影响实际上在信息传递完成后, 复杂网络上结点的信息量组成的向量[6]。

这一节将定义结点相互传递信息的方式。假设一个的概率传播矩阵，中的每个元素代表结点j传播给结点i的信息占结点i所接受的信息比例，公式：

 式(3-1)

其中N是复杂网络结点的个数，是网络中邻接矩阵。该式子意义是当结点j向结点i传递信息时，结点i不会全接受，而是按所定义的比例来接收。用矩阵公式如下：

 式(3-2)

其中，D是由结点的度产生的对焦矩阵，W是复杂网络的邻接矩阵。概率传播矩阵的定义和布朗粒子的跳跃概率定义相似。

3.1.2 TNV算法

假如把结点s投影到向量空间中并形成一个向量，先设置s为源结点，假设它每次都向外发送一个单位的信息。开始时，结点对网络没有任何影响，表达式,表示结点s 一条信息，而其他的结点没有。通过下面的信息传递过程得到结点s对整个网络的影响：

输入：传递的次数T

初始化：，s结点有一个单位的信息，而其它结点没信息。

输出：网络的结点上得信息分布，即结点s在高维空间中的向量表示形式。

（1）计算图邻接矩阵W

（2）计算度的对角矩阵D

（3）计算

（4）：;

（5）如果传递t小于T，返回第四部。否则数据结果为。

表示了结点s对复杂网络影响，该结果就可以作为结点s映射N维空间的向量。

将复杂网络中每个结点作为源结点，按照以上的步骤可以向复杂网络的其他结点传递信息，因此得到每个结点就在高维空间中用向量表示。即就能得到N个高维向量。这N个向量就是网络中N个结点映射到N维空间中的向量。

3.2基于网格和密度聚类算法

本节将在图3-1描述了算法GDCD的主要部分。对于每一个新的数据记录x本节将它映射到一个网格g，然后利用grid\_list来更新*g*的密度。然后，本文周期性的（每隔*gap*时间）形成簇以及移除稀疏网格。下面本文描述确定*gap*的策略，管理活动网格的列表（*list*）以及产生簇。

3.2.1 网格检测以及时间间隔gap

为了挖掘数据流的动态特征，本文在部分3开发的密度网格方案会逐渐地使得每个数据记录和网格的密度变小。如果一个稠密网格长期不接收一个新的数据，它可能会降级成为一个过渡网格或者一个稀疏网格。另一方面，如果一个稀疏网格接受一些新的数据记录，它可以升级成为一个过渡网格或者稠密网格。因此，经过一个时间周期后，每个网格的密度应该被重新检查，簇也要调整。

一个关键的决定是确定检查网格的时间间隔的长度。本文发现这个时间间隔*gap*不能太大也不能太小。如果*gap*太大，数据流的动态变化不能够被很好地识别出来。如果*gap*太小，会导致在离线部分频繁地计算从而增加了负载。当这种计算负载过重，离线部分的处理速度可能不能匹配数据流输入的速度。

本文提出以下的策略来确定*gap*值的合适大小。本文认为使一个稠密网格降级成为稀疏网格所需的的最少时间与使一个稀疏网格成为一个稠密网格所需的时间差不多。为了确保检查除足够频繁地去监测任何网格的密度变化，本文将*gap*设定为这些最小时间的中的最小值。

对任何稠密网格g，从稠密网格成为一个稀疏网格所需的最少时间是：

 式（3-3）

对任何稀疏网格g，从稀疏网格成为一个稠密网格所需的最少时间是：

. 式（3-4）

基于以上两个命题，本文选择足够小的*gap*，这样可以识别一个网格从稠密变为稀疏的任何变化，或者从稀疏变为稠密（的任何变化）。因此，在GDCD中本文设定：

 式(3-3)

3.2.2 检测以及移除稀疏网格

针对基于密度的方案一个严重挑战是大量的网格，尤其是高维数据。比如，如果每一个维度被分为20个部分，那么将有20^d（指数级）个可能的网格。

一个关键的观察是空间中的大部分网格是空的或者只接收数据不是很频繁。在本文的实现方法中，本文只为那些非空的网格分配内存用来存储特征向量，这可以形成一个非常小的网格子空间。不幸的是，实际上，由于在数据错误中形成的离群点数据，导致在聚类过程中不断地增加处理非空网格的时间，这个方法的效率不是足够高。由于这些网格包含非常少的数据，本文称它为稀疏网格。由于大量的数据流高速地到达而且运行很长时间，这样使得稀疏网格不断累积，它们的数目会变得非常巨大，最终导致系统运行越来越慢。因此本文有必要周期性地检测和移除这样的稀疏网格。这在图1中的D-Stream算法中的13行描述。

那些的网格代表稀疏网格。然而，有两种原因会导致一个网格的密度会小于Dl。第一个原因是它接收到非常少的数据，第二个原因是那个网格虽然在过去时间接收到许多数据但是由于受衰减因子的影响它的密度不断变小。只有前面那个原因导致的稀疏网格才是真的，需要移除。有后者原因导致的稀疏网格不应该被移除，因为它包含了许多数据，而且它经常升级，可能会成为过渡网格或者稠密网格。通过大量的实验，本文发现由于后者原因从而导致错误地移除这些网格会严重地恶化聚类的质量。

本文定义了一个密度阈值函数来区分这两种不同类别的稀疏网格。

**定义3.1 密度阈值函数**假设一个网格g最后的更新时间是tg，那么在时刻t（t>tg），密度阈值函数是：

 式(3-4)

命题 函数具有以下一些性质：

(1) 如果 那么

.

(2) 如果那么  ,.

本文从所有稀疏网格中用检测出松散的网格。在周期性检测中，在时刻t，如果满足以下条件，本文判断那个稀疏网格是一个松散网格。

(S1);and

(S2)如果g之前被删除 (在时间) ,

其中>0 是一个常量。

注意和在是特征向量中存储。

在D-Stream中，本文维护一个*grid\_list*，它包括那些考虑用来作聚类分析的网格。*grid\_list*使用可以解决冲突的双链表结构的哈希表实现。哈希表运行检索、更新以及删除。当将每一个网格联系到一个特征向量的入口，哈希表的关键词就相当于一个网格。

本文使用下面的规则从*grid\_list*删除松散网格。

（D1）在图3.1中的10行的周期性检测中，所有满足（S1）和（S2）条件的网格被标识为稀疏，但是以后一直在等待（不做任何操作）直到下一个被认为可以删除的周期性检测。

（D2）在下一个周期性检测中，如果一个被标识为稀疏的网格，自从上一次检测后再也没有收到任何数据，本文将它从格子表中删除。否则，检查g是否满足如上(S1)和(S2)；如果是，本文保持g标识为稀疏不变；如果否，本文将标签重置为正常。

本文应当注意到，一旦一个松散网格被删除，它的密度实际上被重置为零，由于它的特征向量被删除了。如果后来又有新的数据记录映射到上述网格，本文可能需要增加上述的网格，但是它以前的记录被抛弃了，它的密度将从新重零开始。这样一个动态机制可以维护内存中适宜大小的网格，节省了聚类的计算时间，以及阻止了对内存中松散网格无尽的计算。

尽管删除松散的网格对GDCD算法的效率性能非常关键，该方案是否正确，一个重要的问题是这种删除是否会影响到聚类的结果。特别地，由于一个松散的网格可能在以后时间接收到数据从而成为过渡网格或者稠密网格，本文需要知道这种删除是否有可能阻止这个网格被正确地标识为过渡网格或者稠密网格。本文已经设计了一个密度阈值函数和一些删除规则，这样的话过渡簇或者稠密簇不会被错误地删除，由于对松散网格的移除。

考虑到一个网格g，它在时刻t的密度是，假设它在t（每次它的密度被重置为零）之前已经被删除若干次了，因为它的密度在许多时刻比密度阈值函数值要小。假设这些密度值没有被清除而是被全部保留，那么网格g的密度是，本文将称为网格g的完全密度函数。

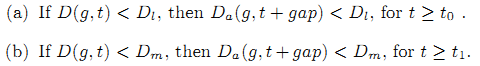
现在本文提出了一些密度函数重要的理论性质，这些性质可以保证GDCD系统的正确功能。本文将要阐明，如果一个网格在后来会成为一个过渡网格或者稠密网格，将它作为松散网格删除将不会影响其以后的升级（就是说松散网格在以后可能升级为过渡网格或者稠密网格）。

本文首先调查的问题是，如果一个网格g作为松散网格被删除，如果它以前从没有过从grid\_list中删除，那么g是否有可能成为非松散的呢？本文将用下面的结论回答这个问题。（答案是g不能是非松散的，一定是松散的）

命题 假设网格*g*作为松散网格被删除的最后时刻是，以及*g*接收一个数据记录的最后时间是**。如果在现在时刻t，本文有，那么本文同样有。

命题非常重要由于它表明了删除一个松散的网格不会导致过渡网格或者稠密网格被错误地删除。它表明，如果网格g在t时刻由于它作为松散网格被删除，，那么即使在以前没有发生过任何的删除（操作），它仍然是一个松散网格，并不能成为一个过渡网格或者稠密网格，因为。

**命题3.2** 一个网格g在时刻t的密度是D(g,t)，而且g在t+1和t+gap之间没有收到任何数据，那么存在以及使得：



命题是一个关键的结论，它使得（S1），（S2），（D1），（D2）一起正确地起作用。它表明，当时间持续足够长，本文将永远不会因为对过去数据的移除而删除一个过渡网格或者稠密网格。如果如果一个网格是稀疏的（非稠密），那么当它被删除时，它肯定是稀疏的（非稠密），即使考虑到过去那些删除的数据。本文知道表示网格的密度（假设在以前从来没发生过删除（操作））。结果表明，在一个初始化阶段后，删除松散的网格不会影响聚类的结果。

|  |
| --- |
| Algorithm **GDCD**(*Network netSet*) |
| **Input:** *netSet*: the complex network  **Output:** *Clustering：every grid has a label；and the label represents the cluster it belongs to;* |
| 1. **procedure GDCD**  2. *tc = 0;*  *3. initialize an empty red-black tree for grid\_list;*  *4.* ***while*** *data stream is active* ***do***  5. transform nodes to vector x =(x1,x2,…,xd);  6. determine the density grid g that contains x;  7. **if**(g not in grid\_list)  8. insert g to grid\_list;  9. **end if**  10. update the characteristic vector of g;  11. **if** tc mod gap == 0 **then**  12. detect and remove sporadic grids from grid\_list;  13. call **generate\_clustering**(grid\_list);  14. **end if**  15. tc = tc + 1;  16. **end while**  17. **end procedure** |

图3-1 基于密度与网格的社团发现聚类算法(GDCD)伪码

伪码说明：图3.1所示的是GDCD的算法伪代码，由于该算法是基于数据流的算法，每次读入数据向量,将每个向量所属的格子存入到Grid\_list中。每次去除“噪声”，并对密集格子做聚类。

3.2.3相邻网格的合并

本文将聚类算法描述成每隔时间gap生成一个聚类（generate\_cluster）。本文用网格之间的引力值来决定网格的合并。在generate\_cluster中，标准基于密度的合并算法是将相邻各自合并来形成聚类。然而在本文的算法中，本文合并两个邻接格子当且仅当他们是强连接的。本文定义两个格子是强连通的当它们的每个方向的引力都是大于一个阈值.即，两个格子g和h是强连通的如果并且。

为了设定的近似值，本文从定理3.3指导所有相邻的格子引力和是逼近。将集合一对格子P集合定义为：

 式（3-5）

定理3.3的结果能被写成：

 式（3-6）

因此，总的引力接近但是不超过而且平均的介于格子对（g,h）平均引力但是不超过, 是大小，能够简单算出每一维分类。同样地定义了一个密格子，本文定义,是一个参数。两个相邻的格子g和h是强连接的如果并且。

合并的过程中，我们使用了一种叫并查集的数据结构，从而提高合并和查找的效率。它主要涉及两个基本操作，分别是：

A.合并两个不想交集合

B.判断两个元素是否属于同一集合

（1）合并两个不相交集合（Union(x,y)）

合并操作很简单：先设置一个数组Father[x]，表示x的“父亲”的编号。那么，合并两个不想交集合的方法是，找到一个集合的祖先根节点，将另外一个集合最久远的祖先的父亲指向它。

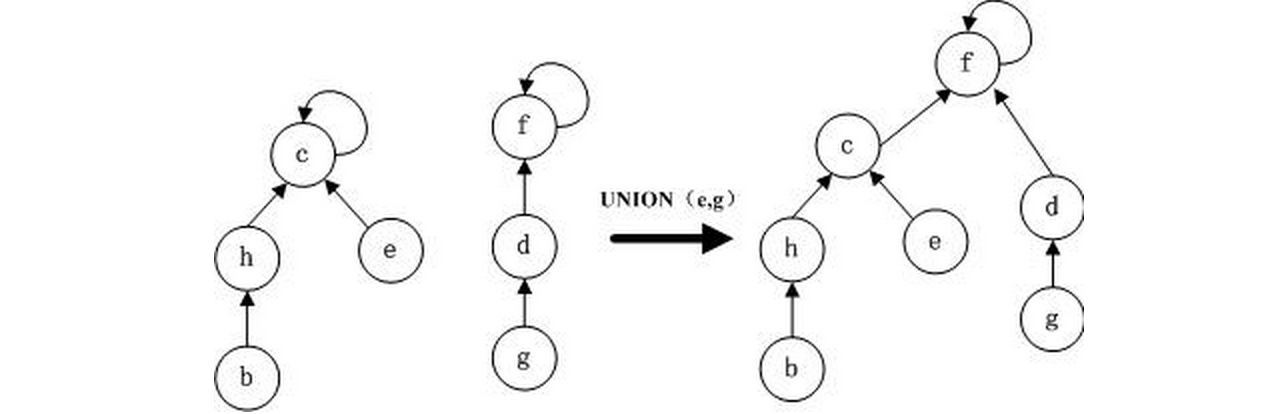


图3-2 聚类网格合并实例

上图3-2为两个不想交集合，b图为合并后Father(b)=Father(g)

(2)判断两个元素是否属于同一集合(Find\_Set(x))

本操作可转换为寻找两个元素的最久远祖先是否相同。可以采用递归实现。

另外我们对并查集进行了优化：寻找祖先时，我们一般采用递归查找，但是当元素很多亦或是整棵树变为一条链，每次Find\_Set(x)都是O(n)的复杂度。为了避免这种情况，我们需对路径进行压缩，即当我们经过”递归”找到祖先根节点后，“回溯”时顺便将它的子孙节点都直接赋值指向祖先，那么以后再次Find\_Set(x)时复杂度就变成O(1)了，如下图3-3所示。可见，路径压缩方便了以后的查找。

在Union(x,y).按秩合并

即合并的时候将元素少得集合合并到元素多的集合中，这样合并之后树的高度会相对较小[7]。

该并查集空间复杂度为O(N)，建立一个集合的时间复杂度为O(1)，N次合并M查找的时间复杂度O(M Alpha(N))，这里Alpha是Acherman函数的某个反函数。

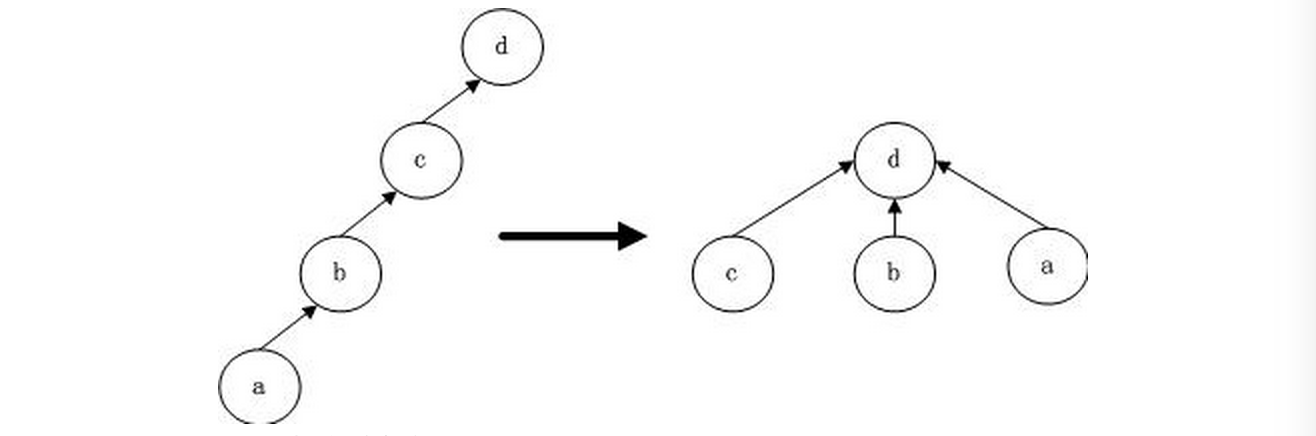


图3-3 网格查找以及优化

|  |
| --- |
| Algorithm Generate\_Clustering(*Grid\_list*) |
| **Input: Grid\_list，a protatial girds for clustering;**  **Output**: ***Clustering;*** |
| 1. **procedure Generate\_Clustering(grid\_list)**  2. update the density of all grids in grid\_list;  3. assign each dense grid to a distinct cluster；  4. label all other grids as NO\_CLASS;  5. **foreach** grids g in grid\_list  6. **foreach** neighboring grid h of g  7. **if** g and h are strongly correlated  8. Union\_clustering(g,h);  9. **end if**  10. **end procedure**  11.  12**. procedure Union\_clustering**(x,y)  13. fx = **find\_cluster\_label**(x);  14. fy = **find\_cluster\_label**(y);  15. **if** fx != fy  16. fx.label = fy;  17. **end if**  18. **end procedure**  19.  20. **procedure find\_cluster\_label**(x)  21. r = x;  22. **while** r.label != r;  23. r = r.label;  24. **end while**  25. **return** r;  26.  **end procedure** |

图3-4 网格合并聚类伪码

伪码说明：该伪码为网格合并过程的伪码，包括3个函数Generate\_clustering()、Union\_Clustering()、find\_cluster\_label()。Generate\_clustering()首先枚举Grid\_list中的每个邻格，然后将强连接的邻格进行合并，合并过程用到Union\_Clustering()函数，在合并中，find\_cluster\_label()对树状结构进行了优化。

# 第四章 实验及分析

4.1实验环境和测试数据

本章对算法进行测试实验，并对其各项性能包括质量和效率和传统算法进行了比较。所有工作都是在主频为2.3GHz，三级缓存为3MB，内存容量为2GB，的双核四线程的笔记本PC上进行的。本文在DEV C++上实现算法并且用Matlab 界面作图。在试验中，本文设定参数=3.0，=0.8，=0.999。

本文用了两个测试数据集。第一个是测试数据集是一个真实的数据集合KDD CUP-99，它包含由MIT林肯实验室收集的网络入侵数据流。这个数据集包含了总共5个聚类，并且每个连接记录包含42个属性。其中有总共34个连续属性被用作聚类。另外，本文也用一些自己合成的数据集来测试算法的可扩展性，合成的数据占内存空间为1M-60M的个数设为4，维度设为2到40.在实验中，本文规范化所有的数据集。每一维甚至被分成很多段，每一段有长度，本文会表明不同的长度得所产生的实验结果。

4.2社团发现的演化过程与效果分析

本文发现，数据流的时序可以对聚类结果产生重大影响。为了验证GDCD算法效能，本文产生合成数据集有两个不同的顺序。

首先，本文随机挑出30K包含4个聚类的二维数据，包括5k的不在聚类内部的即在外围空间的数据。原始数据集的分布见图4-1。这些聚类有些是非凸的而有些是外凸的。本文概括了数据在每一个时间间隔。每次，任何数据点尚未产生被作为新的数据记录同样可能。因此，数据从不同的簇和离群值交替出现在数据流。以GDCD最终测试结果显示在图4-1。本文把。从图4-1本文可以看到，该算法可以发现四类无需用户提供的数字集群。它比采用CluStream自k-means将许多离群数据集不k-均值算法更有效。本文也可以看到，本文的方案检测零星的网格可以有效地去除异常值。

在第二个测试集，本文目的是GDCD算法能够动态的将对待数据的变化并且能够移除实时的外围的数据在调整的过程中。为此，本文将4个类分类并且将它们一个接一个地产生它们。在这个测试中，本文分析177K数据点集包括1K的随机边缘数据。数据分布如图4-1，本文检验聚类结果在三个不同的时间戳，包括=3000=6000=12000。很明显GDCD算法能实时地适应动态的数据流的演变并且免疫噪声。

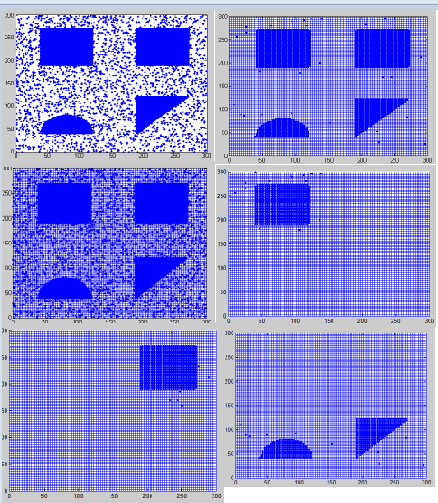


图4-1 聚类数据集，以及聚类的过程

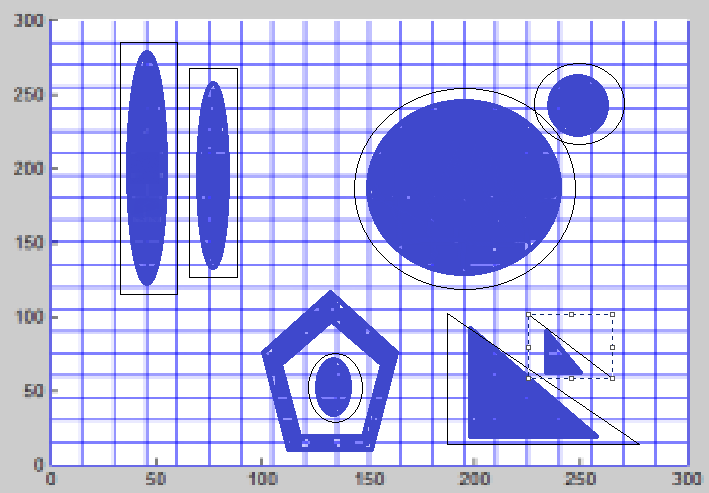
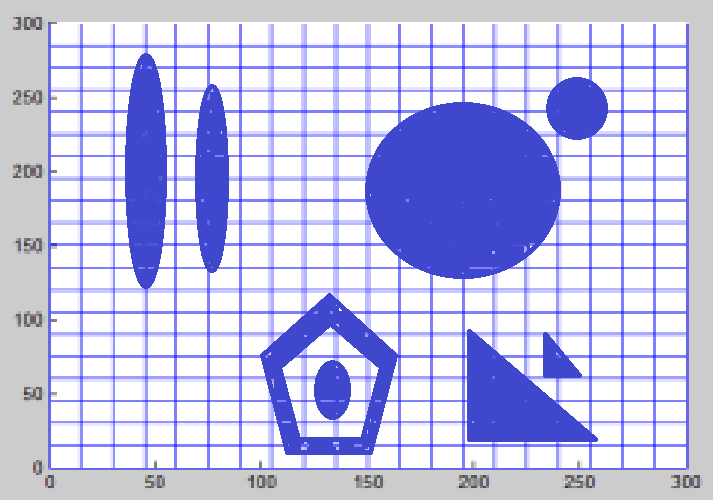
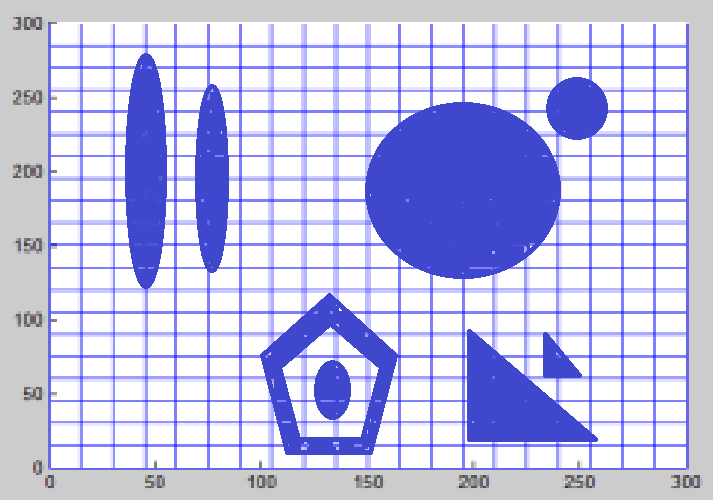
4.3社团发现算法性能比较

本文首先在一个人造数据集比较DS0和DS1来看用网格引力的效果。结果在图7表明。本文看到DS1用了引力能得到更好的聚类。而没有用到引力DS0算法将很多聚类合并到了一起因为它只是用了密度信息而没有用引力。

接下来，本文比较DS0，DS1和CluStream在合成数据，如图4-1和上述KDD CUP-99数据集设置。本文还测试DS0和DS1根据不同的网格粒度。在不同时间的聚类结果的正确率示于图4-2（a）。在图中，len为在规范化的尺寸每个隔开的段的大小。例如，，有50个段中的每个尺寸。从图4-2中，在合成的数据通过GDCD设置平均正确率大于96.5％。本文看到的DS1比DS0显著更好时,len = 0.5，这表明使用格吸引力的优点。当len为较大时，栅极具有一个粗糙的分辨率和使用格吸引力是更突出的优点。

对KDD CUP-99数据的结果。该数据集包含约500万的网络连接。每个连接标记为或者是正常，还是作为攻击，只与一个特定的攻击类型超出四种可能类型。精度正确地标记的连接的比率。从图4-2（b）中，本文看到，DS0和DS1对KDD CUP-99的平均正确率大于92.5％。

本文还比较了由和那些CluStream聚类结果的质量。由于非凸的合成数据组，CluStream不能得到他们正确的结果。因此，其质量无法相比，D-Stream。因此，本文只比较从KDD CUP-99的网络入侵数据两种算法的精度和平方距离（SSQ）的总和。图4-2（c）显示的结果。本文可以看到，这两个DS0和DS1的在不同时间的平均质量总是比CluStream的更好。本文也看到，DS1优于DS0，验证使用格吸引力在基于密度的聚类的有效性。

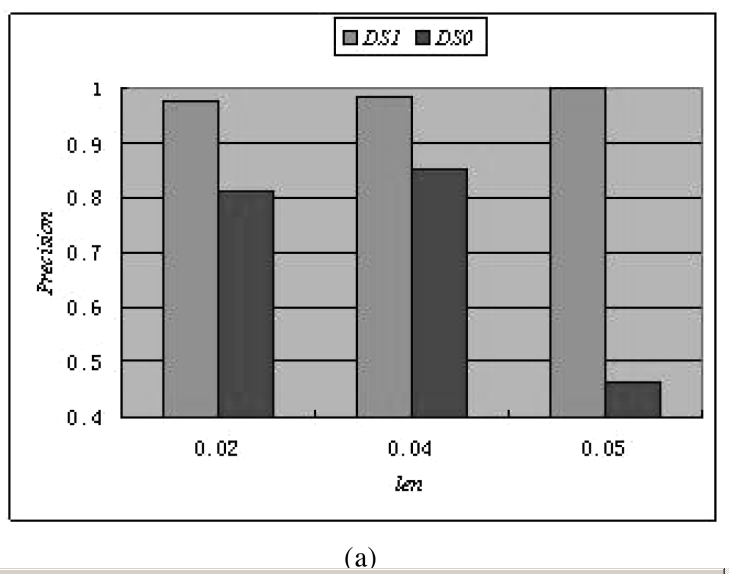


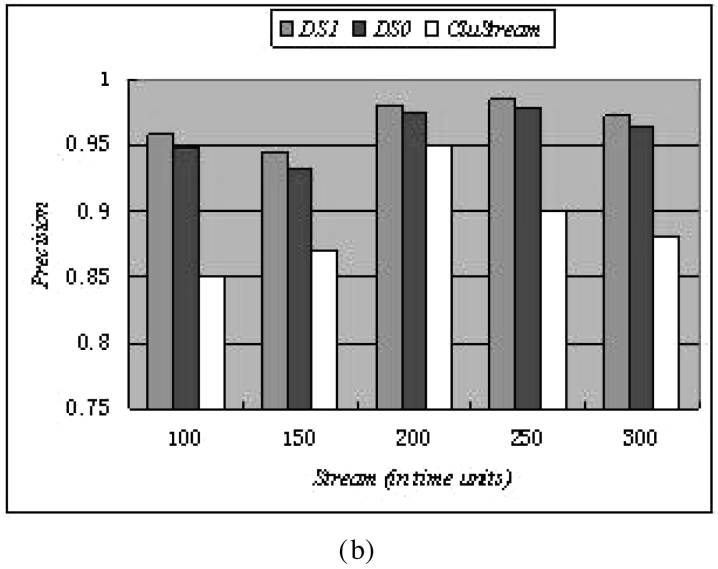
（a） （b） （c）

图4.2基于引力和不基于引力的聚类比较

4-4时间和空间复杂度性能比较

本文测试和比较GDCD和CluStream的聚类速度。首先，两种算法都在KDD CUP-99的数据不同大小的测试。该时间是所用的算法来处理所有的数据的总的时钟时间。对于GDCD，处理数据的在线任务和离线工作的聚类并行执行，以流水线的方式。本文可以看到，这两个版本的GDCD，有着非常密切的时间效率。本文也看到，CluStream需要四到六倍多集群的时间。 GDCD是有效的，因为它只是把每一个新的数据记录到相应的网格的在线组件，而不计算距离为CluStream一样。此外，零星网格的动态检测和删除节省很多时间。它也可以看出，GDCD具有更好的可扩展性，因为它的聚类时间增长较慢具有增加数据大小。本文还比较使用的GDCD，有和没有删除零星格的记忆，本文可以看到，无需拆卸零星的网格，GDCD的内存使用量的增长速度非常快。当零星网格被使用，存储器用法可扩展至非常缓慢。





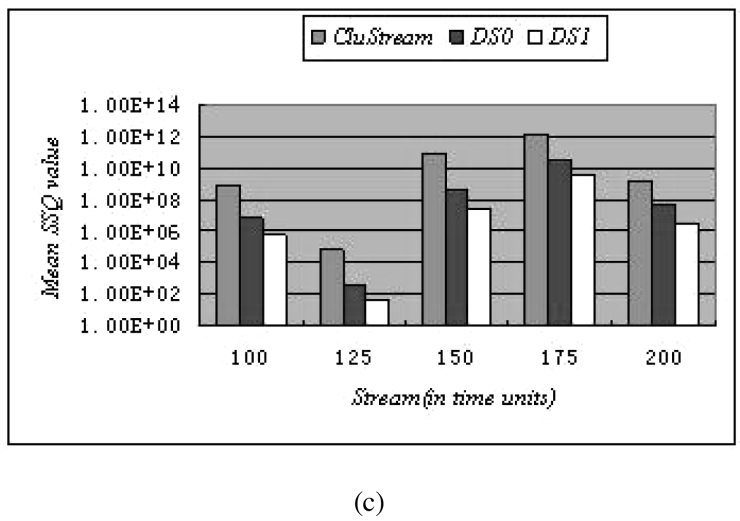


图4.3两种算法的网格长度和准确性的柱状图

接着，所有的算法都是对KDD CUP-99数据具有不同的维数进行测试。本文设定数据设定为100K的大小和变化的维度为2至40。本文列出在不同维数的时间成本所有的算法。 DS0和DS1是3.5到11倍CluStream更快，更好的扩展。

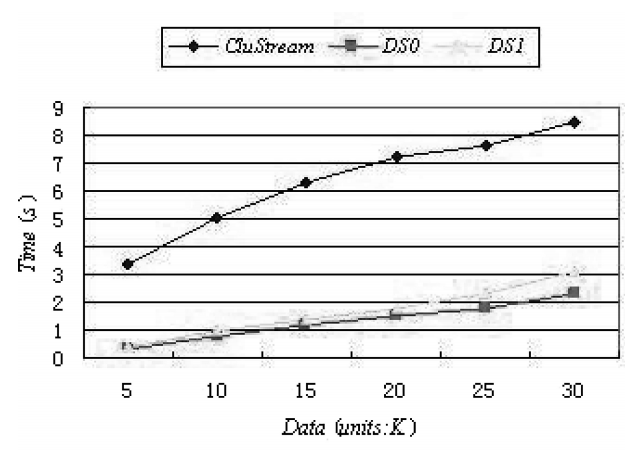


图4.4数据规模和时间的比较

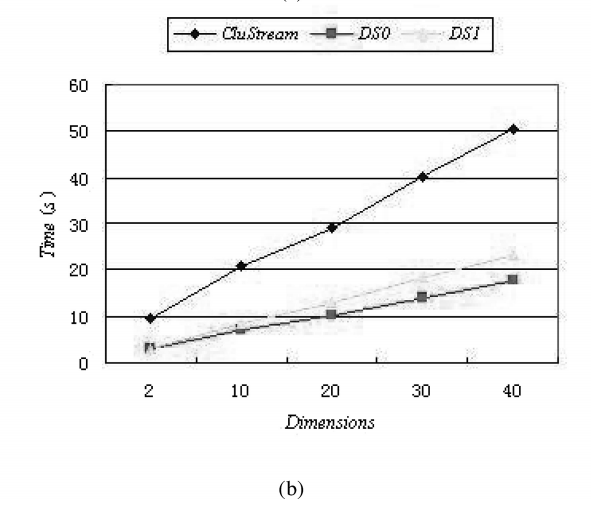


图4.5数据维度和时间的比较

# 

# 致 谢

感谢李雁妮教授的辅导与敦促、理解与包容。做研究艰苦、却又快乐。李老师一丝不苟、严谨求实、踏踏实实的精神，不仅授我以文，且教我做人。感谢马丁同学、姜淳涵同学、李省委学长的共同研究。他们细心辅导我的研究。敬佩老师的专业水平，并且治学严谨和科学研究是我永远学习的榜样，这些品质将积极影响我。同时她不求回报、无私奉献的精神很让我感动，向他们表示由衷的感谢。   
感谢父母，感谢老师，感谢同学，感谢学长学姐！

# 

# 参考文献

[1] 丁金凤.基于网络与密度的数据流聚类算法研究.哈尔滨工程大学,2010

[2] 骆志刚.复杂网络社团发现算法研究新进展.《国防科技大学学报》,2011

[3] 王观玉.基于聚类的复杂网络社团发现算法《计算机工程》,2011

[4] 张鑫.动态复杂网络中增量式社团发现方法的研究与实现. 西安电子科技大学,2013

[5] 杨博.复杂网络聚类方法.《软件学报》,2009

[6] 庞传军.基于聚类的复杂网络中社团发现算法的研究. 青岛大学,2009

[7] 郭莉莉.改进的Lidar数据桥梁提取算法研究.《计算机工程与设计》，2014.

[8] Li Tu & Yixin Chen.stream data clustering based on Grid Density and Attration.南京航空航天大学.华盛顿大学,2009

[9] Jiadong Ren etc.Clustering over Data Streams Based on Grid Density and Index Tree.Yanshan University,2011

[10] Jain A K,Murty M N,Flynn P J.Data Clustering:A Review.ACM computing surveys,1999,31(3):264-323.

[11] Arasu A,Babcock B,Babu S.STREAM:The Stanford Data Stream Mangement system.IEEE Data Engineering Bulletin 2003,26(1):19-26.

[12] Hllerstein J M,Franklin M J,Chandrasekaran S,et al.Adaptive Query-Processing: Technology in Evolution.IEEE data Engineering Bulletin.2000,23(2):7-18

[13] Madden S,Shah M,Hllerstein J M,et al.Continously Adaptive Continuous Queries over Streams.Proc of the ACM SIGMOD International Conference on Management of Data.Madison,Wisconsin,USA.2002:49-60

[14] Krishnamurithy S,Chandrasekaran S,Cooper O,et al.TelegraphCQ:An Architectural Status Report,Buletin of the IEEE computer Society.2003,26(1):11-18.

[15] Jin C,Qian W,Sha C,et al.Dynamically maintaining frequent item over a data Stream.Proc of the 12th ACM Conference on Information and Knowledge Management New Orleans,LA,USA.2003:287-294.