

摘要

科技的进步为我们创造并积累了大量的数据，因此我们需要更多的有效的方法来挖掘不同形式的数据中的有用的信息，这也是数据挖掘的核心任务。针对日益增加的轨迹数据，这篇文章的第一部分首先提出了一个衡量轨迹数据的相似度方法，它的基本想法是为每条轨迹构建一个行为网络 (*Mobility Network*)，用行为网络的特征来反映轨迹的时间和空间信息。通过比较轨迹对应的行为网络的特征，例如节点出现次数的分布、访问节点的时间的分布，进而我们可以得到轨迹的相似度。在第二部分中，这篇文章提出了一个新的图聚类算法 (*Attractor*)，其基本思想是将一个网络看作一个动态系统，通过节点与节点之间的交互，处于同一个社团内的节点之间距离将会减小，处于不同社团的节点之间距离将会变大，通过检测节点与节点之间的距离，最终我们可以得到这个网络的社团结构。大量的实验说明了这个算法的优势。在上述两个算法的基础上，我们以一个出租车数据集为例，完成了对出租车轨迹的聚类分析。

关键词：数据挖掘，聚类，社团挖掘，相似度度量，轨迹

ABSTRACT

With the prevalence of technologies in our lives, much more data are generated and stored. So we need to develop more effective and efficient methods to acquire the knowledge behind the data. And this is the core task of data mining. The first part of this paper proposes a similarity measure for trajectory data. Its basic idea is to construct a *Mobility Network* for each trajectory, in which nodes represent the locations appeared in the trajectory and edges record the move from one place to another place. We can measure the similarity of two trajectories by comparing their networks like the weight distributions of nodes. In conjunction with the clustering method proposed in the next part, we make an analysis on a taxi data set. In the second part of this paper, we develop a new graph clustering algorithm (*Attractor*), which works on metric space. The basic idea is to envision the target network as a dynamical system, where each node interacts with its neighbors. The interaction will change the distances among nodes, while the distances will affect the interactions. We will see that nodes sharing the same community move together, thus revealing the community structure automatically. Extensive experiments demonstrate that our algorithm allows the effective and efficient community detection and has good performance compared to state-of-the-art algorithms.

Keywords: data mining, clustering, community detection, similarity, trajectory

目 录

第1章 绪论	1
1.1 引言—从数据中挖掘知识	1
1.2 研究背景与相关工作回顾	2
1.2.1 轨迹分析介绍	2
1.2.2 聚类分析介绍	3
1.3 本文组织结构与章节安排	5
第2章 轨迹数据相似度度量	6
2.1 基本思想与主要贡献	7
2.1.1 基本思想	7
2.1.2 主要贡献	8
2.2 相关工作回顾	8
2.2.1 基于动态时间规整的相似度度量	9
2.2.2 基于最长共同子串的相似度度量	10
2.2.3 基于最小编辑距离的相似度度量	10
2.3 基于行为网络的轨迹相似度度量	11
2.3.1 准备工作	11
2.3.2 基于行为网络的轨迹相似度度量	12
2.3.2.1 将轨迹映射到行为网络	12
2.3.2.2 行为网络相似度比较	13
2.4 实验	14
2.5 总结	16
第3章 基于动态交互的社团挖掘算法	17
3.1 基本思想与主要贡献	18
3.1.1 基本思想	18
3.1.2 主要贡献	18
3.2 相关工作回顾	19
3.3 基于动态交互的社团挖掘算法	20

3.3.1 准备工作	20
3.3.2 交互模型	20
3.3.3 Attractor 算法.....	23
3.3.3.1 Attractor 算法	23
3.3.3.2 对耦合参数 λ 的讨论	24
3.3.4 时间复杂度分析	24
3.4 实验	25
3.4.1 人工数据集上的对比试验.....	25
3.4.1.1 改变噪声边比例	26
3.4.1.2 改变网络稀疏度	26
3.4.2 真实数据集上的对比试验.....	27
3.4.2.1 社团结构已知的网络	27
3.4.2.2 社团结构未知的网络	29
3.4.3 社团规模分布和噪声点检测.....	30
3.4.3.1 社团规模比较.....	30
3.4.3.2 噪声点检测	31
3.4.4 时间复杂度测试	31
3.5 总结	32
第4章 出租车GPS轨迹聚类分析.....	34
4.1 数据集说明	34
4.2 聚类方法说明	34
4.3 结果及讨论	35
第5章 结语	36
5.1 本文工作总结	36
5.2 下一步工作	37
参考文献	38
致 谢	41
附录 A Attractor 算法	42
A.1 Attractor 算法伪代码标示	42
附录 B 轨迹相似度度量	43
B.1 出租车轨迹相似度完整结果	43

第1章 绪论

1.1 引言—从数据中挖掘知识

科技的发展使得越来越多的各种各样的数据被产生、纪录、保存下来。我们希望通过对这些数据的研究，得到一些有用的信息，例如群体的行为模式、传染病的传播方式等等，进而帮助人们做出好的决策^[1]。因此，各种各样的方法被先后提出，来帮助人们发现数据中隐藏的有用的信息^{[2][3]}。

这篇文章中，我们将要介绍两个基础算法，并将这两个新的算法应用在实际数据中，从而尝试从数据中挖掘有用的知识。具体来说，在这篇文章中，我们以聚类分析为中心，以轨迹数据(行为数据)为应用对象，首先提出了度量轨迹之间相似度的方法，在此基础上，用一个新提出的聚类算法对这些轨迹进行聚类。但是，在具体介绍我们的工作之前，首先介绍一下聚类分析的背景以及流程。

通常，从数据中学习知识包含以下流程^{[4][5]}：

- **数据采集** 首先需要采集数据，包括确定数据种类、范围等
- **数据预处理** 这个过程包括对原始数据的处理，包括数据整合、去除噪声等
- **数据表示** 对数据进行完预处理后，需要决定数据合适表示，例如特征筛选等
- **数据挖掘** 这个过程中，人们采用各种方法，例如聚类、分类、关联规则分析等方法来发掘数据中的有用的信息
- **结果解释与评估** 最后，需要对得到的结果进行解释与评估，有可能需要重新进行挖掘

其中，**数据挖掘**是从数据中学习知识的最关键的步骤，因此很多时候，数据挖掘泛指从数据中学习知识。数据挖掘主要包含以下方法：

- **分类** 分类的目的是为新数据确定它的类别。一般来说，分类首先学习出目标函数，然后用学到的目标函数来预测新来的未知数据点的类别。
- **聚类** 聚类的目的是将数据分为许多类，使得同一类中的数据非常相似，不相似的数据分布在不同的类中。常见的聚类方法有 k -means, *spectral clustering* 等方法。

- **关联规则分析** 关联规则分析的目的是从数据中发现经常出现的模式，一个经典的例子是人们从超市的大量销售记录中发现买尿布的人也常常买啤酒。经典的关联规则分析方法有：*Apriori*^[6], *FP-growth*^[7] 等。
- **奇异点检测** 奇异点检测的目的是发现数据集中存在的奇异点(与大多数点不相似的少数数据点)。

聚类作为数据挖掘中一项关键手段，在许多领域都有重要的应用^[8]。这篇文章主要关注数据挖掘中的聚类方法特别是图聚类(社团挖掘)，针对现有的图聚类面临的一些挑战，本文提出了基于动态交互的新的社团挖掘方法，并且将聚类算法用于日益重要的轨迹数据，通过实验分析了出租车存在的常见模式。

1.2 研究背景与相关工作回顾

这一小节中，我们从轨迹分析以及聚类分析这两个方面回顾与这篇文章相关的研究背景以及相关工作。

1.2.1 轨迹分析介绍

随着移动设备的普及，越来越多的轨迹数据被记录和保存下来。从这些轨迹数据中，我们可以发现行为的特征以及模式，在轨迹中发现的有用信息可以帮助人们做出更好的决策^①。因此，轨迹数据分析正在吸引越来越多的注意。对于这些轨迹数据，一个关键的问题是如何挖掘这些轨迹数据中的模式与规律。

目前，许多算法借鉴关联规则分析中的频繁项挖掘的思想来挖掘轨迹中存在的模式^{[9][10]}。它们将轨迹挖掘看作在时间约束下的频繁项挖掘，进而找到轨迹中经常出现的模式。下面，简单地介绍一个经典的工作^[10]。

在2007年的一篇文章中^[10]，作者将时间约束引入频繁项集挖掘来寻找轨迹中经常出现的模式。作者用含时序列^②来表示轨迹 T ：

$$T = s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_n} s_n$$

即 $T = (S, A)$ ，其中空间序列 $S = \langle s_0, s_1, \dots, s_n \rangle$ ，时间序列 $A = \langle \alpha_1, \dots, \alpha_n \rangle$ 之后，作者首先定义了空间序列的包含(\preceq_N)：对于空间序列 $S = \langle s_0, s_1, \dots, s_n \rangle$ ，记

① 例如，通过研究道路上行人以及车辆的轨迹，交通部门可以更好地调节城市中交通；社交网络平台可以寻找与某个用户有相同出行模式的其它用户，进而更加精准地推荐好友等等

② Temporally Annotated Sequences(TAS)

另外一个含时序列 $T = \langle (x'_0, y'_0, t'_0), \dots, (x'_n, y'_n, t'_n) \rangle$, $N(x, y)$ 表示 (x, y) 这个点的邻域, 那么记 S 被 T 空间包含 ($S \preceq_N T$) 当且仅当存在 $0 \leq i_0 < \dots < i_k \leq n$, 使得对 $\forall 0 \leq j \leq k, (x_j, y_j) \in N(x'_{i_j}, y'_{i_j})$.

含时序列的“包含” ($\preceq_{N,\tau}$): 对于一个含时序列 T , 给定一个时间范围的阈值 τ , 那么对于一个给定的含时序列 $(S, A) = (x_0, y_0) \xrightarrow{\alpha_1} (x_1, y_1) \xrightarrow{\alpha_2} \dots \xrightarrow{\alpha_n} (x_k, y_k)$ 和另外一个含时序列 $T' = \langle (x'_0, y'_0, t'_0), \dots, (x'_n, y'_n, t'_n) \rangle$, 定义 (S, A) 被 T' 包含 ($(S, A) \preceq_{N,\tau} T'$) 当且仅当 (S, A) 和 T' 满足以下两个条件:

1. $S \preceq_N T'$
2. $\forall 1 \leq j \leq k, |\alpha_j - \alpha'_j| \leq \tau$, 其中 $\alpha'_j = t'_j - t'_{j-1}$

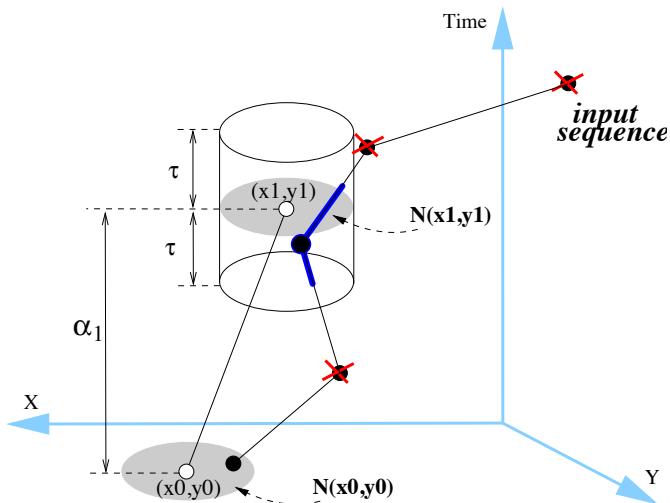


图 1-1 含时序列“包含”示意图, 图片来自[10]

于是对于每一条轨迹模式, 我们就可以计算它在数据集中的支持度, 因此挖掘这个轨迹数据集中常见模式的问题就转化为了频繁项挖掘的问题, 进而作者使用类似 *FP-growth*^[7] 的算法来处理, 这里不再具体描述。

1.2.2 聚类分析介绍

下面将要介绍聚类分析的基本流程, 包括对象之间相似度度量、聚类算法和结果分析检验等, 并且回顾了现有的相关工作。

对于给定的数据集, 聚类分析的目的是将相似的对象分到一类, 不相似的对象分到不同类。为了衡量对象之间的相似性, 首先需要定义给定数据集中各个对象的相似度指标或是距离。两个物体之间的相似度越高, 那么它们之间的距离则越近, 如果两个物体之间的相似度低, 那么他们距离则远。

许多数据可由由 n 维空间中的向量表示, 对于这类数据, 常常可以用 n 维空间的距离函数来刻画对象与对象之间的距离, 常见的距离函数有: 欧式距离、 L_p 范数、曼哈顿距离等。假如距离函数 $dist()$ 选用 L_p 范数, 则对于给定数据集 D 中的两个 n 维向量 $x = x_1, x_2, \dots, x_n$ 和 $y = y_1, y_2, \dots, y_n$, x 和 y 之间的距离 $dist(x, y)$ 可以表示为:

$$dist(x, y) = \sqrt[p]{\sum_{i=1}^d (x_i - y_i)^p} \quad (1-1)$$

对于复杂的数据类型, 例如神经纤维^[5]、行为轨迹等等, 由于 n 维空间中的矢量并不能有效地表示这些对象, 因此度量这些对象之间的相似度并不是一件简单的工作。例如后文中, 为了对轨迹进行聚类, 我们提出了度量轨迹相似度的一个指标, 相关的距离指标的回顾见2.2.

确定对象之间的相似度之后, 需要用聚类算法对这些对象进行聚类。常见的聚类算法有: *k-means*, *DBSCAN*, *Spectral Clustering*, 以及用于图聚类(社团挖掘)的 *Ncut*, *Louvain* 和 *MCL* 等. 在3章中的3.2 小节中, 我们将会详细介绍与本文关系密切的经典的图聚类算法。

完成聚类之后, 还需要对结果进行检验和解释。对于有标签的数据^①, 经典的衡量聚类结果的指标有: *NMI*, *ARI* 和 *Purity*^②. 例如, 假设数据集有 N 个对象, 用 $X = x_1, x_2, \dots, x_N$ 表示用算法得到的这些对象的类别(x_i 代表第 i 个对象的类别), $Y = y_1, y_2, \dots, y_N$ 表示得到的这些对象的真实标签, 那么得到的结果的 *NMI* 指标为:

$$NMI(X, Y) = \frac{I(X; Y)}{[H(X) + H(Y)]/2} \quad (1-2)$$

其中

$$\begin{aligned} I(X; Y) &= \sum_k \sum_j P(x_k \cap y_j) \log \frac{P(x_k \cap y_j)}{P(x_k)P(y_j)} \\ &= \sum_k \sum_j \frac{|x_k \cap y_j|}{N} \log \frac{N|x_k \cap y_j|}{|x_k||y_j|} \end{aligned}$$

① 已经知道数据集的类信息

② <http://nlp.stanford.edu/IR-book/html/htmledition/evaluation-of-clustering-1.html>

$$\begin{aligned} H(X) &= - \sum_k P(x_k) \log P(x_k) \\ &= - \sum_k \frac{|x_k|}{N} \log \frac{|x_k|}{N} \end{aligned}$$

以上就是聚类分析的整体流程，在后文中，为了完成轨迹数据的聚类分析，我们将首先定义针对轨迹数据的相似度指标，在此基础上将轨迹聚类问题转化为了图聚类问题；然后利用提出的*Attractor* 算法完成了对轨迹数据的聚类。

1.3 本文组织结构与章节安排

这篇文章的核心工作是提出了轨迹数据的相似度衡量指标(2章)，以及提出了一个新的社团挖掘(图聚类) 算法(3章)。之后在第4章中，我们将相似度度量以及图聚类这两个算法应用于出租车轨迹数据上，完成了对轨迹数据的聚类。具体章节安排如下：

1. 第一章主要介绍了研究背景，包括数据挖掘的有关概念和这篇文章的研究背景及相关工作回顾，并介绍了文章的组织结构
2. 第二章主要提出了一个衡量轨迹数据相似度的方法，并在出租车数据集，分析了用这个相似度指标得到的结果
3. 第三章介绍了一个新的图聚类算法，并且通过大量的与经典算法的实验，展示了我们的算法的优势
4. 第四章展示了轨迹聚类的结果，具体来说，在这一章中，对于一个出租车数据集，我们首先用第二章中的方法计算得到它们之间的相似度，在此基础上，将第三章中的聚类方法应用于这些数据，完成了对出租车的聚类。
5. 第五章总结了这篇文章的主要工作，并以此为基础，对接下来的工作进行了展望。

第2章 轨迹数据相似度度量

衡量对象与对象之间的相似度指标十分重要^[11]，因为它许多问题中都有广泛的应用^[12]。例如前面已经提到的聚类算法，所有的聚类算法都有一个使用的前提，那就是必须先存在一个衡量对象与对象之间相似度(距离)的指标；再比如当我们需要对某个事物做出预测时(例如推荐系统、链路预测、空间位置预测等等)，一种常见的方法是通过合适的相似度指标来找到与这个事物最为相似的几个其它事物，然后根据与它相似的其它事物的特征(行为)来预测这个事物将要出现的特征^[11]。

对于可以用 n 维空间中的向量表示的对象，我们可以用欧式距离衡量它们之间的相似度。但是在日常生活中，我们常常面对的是更加复杂的数据类型，例如复杂网络、时间序列等等，这些复杂的数据集上的相似度度量依然是公开的问题。特别的，GPS设备的广泛应用，使得越来越多的轨迹数据被保存下来，挖掘这些轨迹数据中存在的人们活动的规律有着非常重要的应用，而一个有效的度量轨迹数据相似度的指标是解决许多问题的基础(例如聚类、预测)。

在这篇文章中，我们提出了一个新的度量轨迹数据^①相似度的指标：基于空间分布和空间分布的轨迹相似度指标。我们将会发现，这个相似度指标不仅仅适用轨迹数据，它同样也适用与其它行为数据^②，例如点击网页的行为数据。

这一章剩余内容的结构安排如下：2.1 将会介绍将要提出的相似度指标的基本思想以及主要贡献；2.2 简单地回顾了几个经典的衡量轨迹数据相似度的方法；2.3 将会具体地介绍我们提出的衡量轨迹相似度的方法；2.4 是在一个出租车GPS数据集的相似度的实验；最后2.5 节是对我们工作的总结。

① 事实上，这个相似度指标不仅适用于轨迹数据(例如由GPS采集到的轨迹)，也适用于其它行为数据，见下文介绍

② 这里所说的行为数据指的是可以用类似 $\{< t_1, behavior_1 >, < t_2, behavior_2 >, \dots, < t_n, behavior_n >\}$ 这种集合表示的数据。例如轨迹数据可以表示为 $\{< t_1, location_1 >, < t_2, location_2 >, \dots, < t_n, location_n >\}$

2.1 基本思想与主要贡献

2.1.1 基本思想

为了衡量轨迹之间的相似度(距离)，非常重要的一点是找到它们之间的差异，也就是哪些属性使得这些轨迹成为了不同的轨迹。从本质上来说，轨迹数据由时间信息以及空间信息构成，因此我们从空间特征和时间特征这两个角度来考虑轨迹之间的相似度。具体来说，对于每一条轨迹数据，我们先将其映射到一个行为网络(*MobNetwork*)^{[13][14]}，并且根据轨迹数据的空间信息和时间信息，为行为网络赋予一些属性(例如节点的权值和节点的访问时间分布)，使得映射后的行为网络的属性可以反映对应的轨迹的空间特征以及时间特征，然后通过比较行为网络(*MobNetwork*)的差异(相似度)，进而确定这些行为网络对应的轨迹之间的相似度。

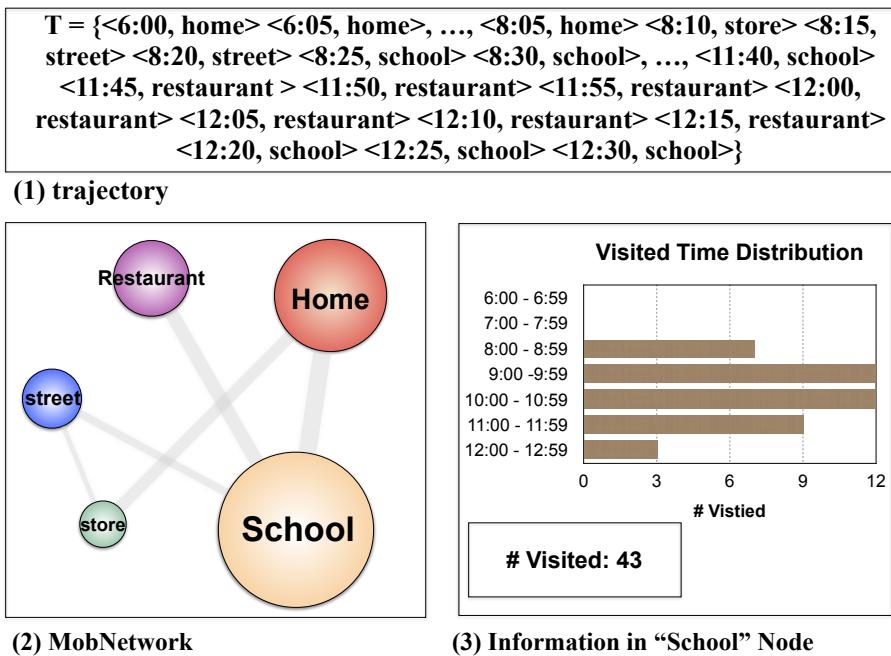


图 2-1 行为网络示意图

图2-1 展示将轨迹(2-1(1)) 转化为行为网络(2-1(2)) 的过程。其中图2-1 (1) 是一个学生从早上6:00到中午12:30的轨迹，时间间隔为5分钟，图2-1 (2) 是这条轨迹对应的行为网络，网络中的每个节点对应一个在轨迹中出现的地点，网络中的边对应地点之间的转移。图2-1 (3) 是每个节点存储的信息的示意图^①，每个节点存

^① 以“School”节点为例

储关于这个地点的两个信息^①: 这个地点在轨迹中出现的次数(作为节点的权重)以及这个地点被访问的时间的分布, 这两者分别反映了轨迹的空间信息和时间信息。于是, 比较轨迹的相似度的问题就转化为了如何比较网络的相似度。很自然地, 对于两个行为网络, 我们可以计算节点权重的分布之间的差异以及对应节点访问时间的分布的差异, 前者反映了两个轨迹的空间的相似度, 后者反映了两个轨迹的时间的相似度, 将这两者结合起来即可得到两个轨迹的相似度。

2.1.2 主要贡献

通过轨迹的时间信息以及空间信息构建的行为网络可以有效地反映轨迹之间的相似度。特别地, 这种相似度指标有以下几个优势:

- 有效地提取轨迹的空间和时间信息:** 将轨迹数据转化为行为网络, 并根据轨迹数据的时间和空间信息赋予行为网络对应的属性, 这个过程可以有效地提取轨迹的空间和时间信息, 并且为之后利用这些空间和时间信息来计算轨迹的相似度做好了准备工作, 在此基础上计算轨迹的相似度变得非常容易。
- 可扩展性:** 首先, 基于行为网络的相似度度量具有较低的时间开销, 因为将轨迹数据转化为行为网络后, 我们只需要通过比较行为网络的属性来确定相似度即可, 而不再需要单独考虑轨迹数据中的每个点。其次, 当行为数据具有更多特征时(我们希望考虑轨迹数据更多的特征时), 通过2.3节介绍的方法, 我们非常容易将这些新特征加入进来, 这点在之后的介绍中可以体现出来。
- 对噪声点不敏感:** 把轨迹数据转化为行为网络后, 不仅仅可以简化运算, 这种方法还可以使噪声点对相似度计算的影响减小。这是由于将轨迹转化为行为网络之后再计算相似度时, 用到了轨迹的时间和空间的整体的特征, 而不是直接考虑轨迹中的每个点, 而少量的噪声点并不会对轨迹整体的时间和空间特征造成影响, 因此可以减少数据中存在的噪声的影响。

2.2 相关工作回顾

轨迹数据的聚类、分类、预测在许多领域都具有重要的研究价值, 而相似度度量是这些工作的基础。由于轨迹数据可以看作时间序列, 因此一些应用于时间

^① 目前的相似度度量方法是比较粗略的版本, 接下来, 我们将要加入边的信息(地点与地点之间的转移), 还进一步刻画轨迹的局部特征

序列相似度度量的方法也可以被用于轨迹数据，下面，我们就来回顾一下这些方法。

2.2.1 基于动态时间规整的相似度度量

动态时间规整^{①[15] [16]}，被经常用来比较时间序列的相似性(距离)。它的基本想法是为两个时间序列找到“最好的”匹配，为了实现这一点，时间序列在时间轴上被拉长或压缩(“Warping”)^[17]，之后两个时间序列上的点再“最合适地”相互匹配。

假设两个时间序列 X 和 Y 的长度分别为 m 和 n ，则 X 和 Y 可以表示为：

$$X = (x_1, x_2, \dots, x_m) \quad (2-1)$$

$$Y = (y_1, y_2, \dots, y_n) \quad (2-2)$$

时间序列规整的目标是找到 X 和 Y 的最佳的匹配路径 W ，记 W 为

$$W = (w_1, w_2, \dots, w_K) \quad (2-3)$$

K 为匹配路径 W 的长度， K 满足条件 $\max(m, n) < K < m + n - 1$ ；匹配路径 W 中的每个元素 $w_k (0 \leq k \leq K)$ 表示 X 中的某个点 x_i 与 Y 中的某个点 y_j 匹配到了一起，记作 $w_k = (i, j)$ ， W 应满足以下约束：

1. 边界条件： $w_1 = (1, 1)$ and $w_K = (m, n)$ ，即 X 的第一个点和 Y 的第一个点要匹配到一起， X 的最后一个点和 Y 的最后一个点要匹配到一起。
2. 匹配顺序：对于 W 中的任意两个点 $w_k = (i, j)$ 和 $w_{k+1} = (i', j')$ ，要求 $i' \geq i$ 并且 $j' \geq j$ 。实际上，这个约束限制了 X 和 Y 只能线性地(在时间轴上拉长或缩短)匹配。
3. 连续性：对于 W 中的任意两个点 $w_k = (i, j)$ 和 $w_{k+1} = (i', j')$ ，要求 $i' - i \leq 1$ 并且 $j' - j \leq 1$ ，这个约束要求 X 和 Y 中的每个点都有匹配。

满足以上约束的匹配有许多可能，动态时间规整是要找出其中“最好”的匹配，也就是使 X 和 Y 距离最小的匹配：

$$dtw(X, Y) = \min\left(\sum_{k=1}^K d(w_k)\right) \quad (2-4)$$

$d(\cdot)$ 是距离函数，例如 $d(w_k) = d(x_i, y_j) = |x_i - y_j|$ 。

^① Dynamic Time Warping (DTW)

在求解两个时间序列的DTW 距离时，动态规划技术常常被用来加快寻找最佳匹配的速度^[18]。

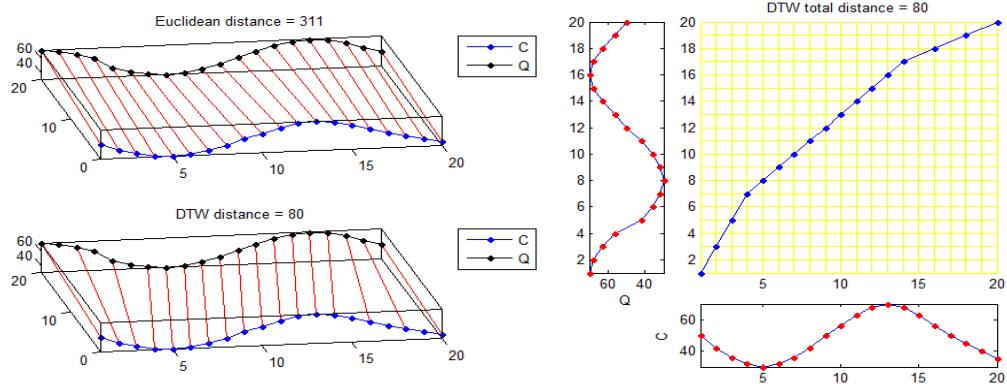


图 2-2 基于DTW的轨迹相似度示意图^[19]

2.2.2 基于最长共同子串的相似度度量

在实际情况中，受设备精度的限制、信号传输过程中存在的噪声以及许多其它可能的情况的影响，实际到的轨迹数据采样点可能与正确值存在着误差，而基于最长共同子串^① 的相似度度量是一种对噪声比较不敏感的方法^[12]。它的基本思想是用两个序列都出现的最长的共同子序列的长度来衡量这两个序列的相似度^[20]。对于包含经纬度的轨迹，需要指定一个阈值 ϵ ，当两个轨迹中的采样点的距离小于 ϵ 时，则认为这两个点可以匹配在一起，即这个点属于这两个序列的共同子序列。通过比较两个序列的最长子序列，进而确定这两个序列的相似度。

2.2.3 基于最小编辑距离的相似度度量

最小编辑距离^② 是一种衡量两个字符串相似度的方法，基于最小编辑距离的相似度度量方法被应用在自然语言处理^[21]，DNA 分析^[22]等领域。它的基本思想是用将一个序列转化为另一个序列所需要的最小编辑代价来衡量这两个序列的不相似程度，即把将一个序列转化为另一个序列需要的最小的编辑操作的代价看作这两个序列的距离。许多情况下，允许的编辑操作包含：将一个字符替换成另一个字符，插入一个新字符以及删除一个字符，并且这三种操作可以有不同的代价^③。

① Longest common subsequence (LCSS)

② Edit distance

③ Levenshtein 开始提出edit distance 这三种操作的代价是一样的

维基百科中的一个例子展示了怎样求两个序列(以“kitten” 和“sitting”两个序列为例, Levenshtein 距离为3) 的最小编辑距离^①:

1. kitten → sitten (将“k” 替换为“s”)
2. sitten → sittin (将“e” 替换为“i”)
3. sittin → sitting (在结尾加“g”)

在实际应用中, 最小编辑距离也通常采用动态规划的方法来得到。

2.3 基于行为网络的轨迹相似度度量

在介绍相似度度量之前, 我们先约定几个之后会用到的定义。

2.3.1 准备工作

定义一 (轨迹) 下文中用 \mathcal{T} 表示轨迹构成的集合, 用 T 表示某一条轨迹, \mathcal{T} 中的第 i 条轨迹用 $T^{(i)}$ 表示。其中, $T^{(i)}$ 由一系列包含时间位置信息的点构成, 即 $T^{(i)} = \{ \langle t_1^{(i)}, loc_1^{(i)} \rangle, \langle t_2^{(i)}, loc_2^{(i)} \rangle, \dots, \langle t_N^{(i)}, loc_N^{(i)} \rangle \}$ (假设这条轨迹有 N 个采样点), 当位置信息包含两个维度时(例如经度和纬度), $T^{(i)}$ 可写为 $T^{(i)} = \{ \langle t_1^{(i)}, x_1^{(i)}, y_1^{(i)} \rangle, \langle t_2^{(i)}, x_2^{(i)}, y_2^{(i)} \rangle, \dots, \langle t_n^{(i)}, x_n^{(i)}, y_n^{(i)} \rangle \}$.

定义二 (含时行为数据) 对于含时间的行为(动作)构成的行为数据, 可以类似地表示。用 \mathcal{B} 表示行为数据构成的集合, 用 B 表示某一个行为数据, \mathcal{B} 集合中的第 i 个行为用 $B^{(i)}$ 表示。其中, $B^{(i)}$ 由一系列包含时间行为信息的点构成, 即 $B^{(i)} = \{ \langle t_1^{(i)}, behavior_1^{(i)} \rangle, \langle t_2^{(i)}, behavior_2^{(i)} \rangle, \dots, \langle t_N^{(i)}, behavior_N^{(i)} \rangle \}$. 根据不同需要, 可以将 $behavior$ 换成某种具体的行为, 比如当把 $behavior$ 换成访问的地理位置时, 这个行为数据就等同于了上面介绍的轨迹数据。

定义三 (行为网络) 对每一个行为数据 B , 我们可以将它映射到一个行为网络 $N = (V, E, PV, PE)$, 其中 V 是节点构成的集合, 对于 V 中的任意一个节点 $\forall v \in V$, v 代表一种在 B 中出现过的行为; E 是边构成的集合, 对于 $\forall e = u, v \in E$, e 表示发生过从行为 u 转移到行为 v , PV 是网络中节点属性的集合, PE 是网络中边属性的集合。

^① http://en.wikipedia.org/wiki/Edit_distance

定义四 (Bhattacharyya Distance^①) 巴氏距离是一种被广泛使用的度量离散变量或者连续变量概率分布之间的距离的指标，它要求随机变量有相同的值域。假设随机变量^② p 和 q 的值域空间为 X , p 和 q 的巴氏距离定义如下：

$$D_B(p, q) = -\ln(BC(p, q))$$

$$where : BC(p, q) = \sum_{x \in X} \sqrt{p(x)q(x)} \quad (2-5)$$

2.3.2 基于行为网络的轨迹相似度度量

2.3.2.1 将轨迹映射到行为网络

将轨迹数据首先映射到行为网络可以带来三个好处，首先是使得轨迹数据相似度度量变得比较容易，因为通过行为网络这种形式，我们可以有效地通过行为网络的属性来把握轨迹的时间和空间信息，进而比较轨迹的相似度；其次是降低了复杂度，因为我们不用直接研究轨迹中的每个数据点；并且使得相似度计算对噪声不敏感，因为计算相似度时，我们用到的是轨迹的时间和空间的统计信息，而少量的噪声点并不会对轨迹整体的时间特征和空间特征造成较大的影响。

接下来，我们需要解决一个关键问题：怎样比较好地将轨迹映射到行为网络上，也就是使得行为网络尽可能充分地保留并反映轨迹的时间信息和空间信息，并且方便之后计算相似度。对于这个问题，我们可以这样做：将轨迹中出现的地点的集合作为网络中的点集，将轨迹中出现的一个地点到另一个地点的转移作为网络中的边集，然后通过赋予的网络的属性来体现轨迹的空间和时间特征。对于GPS数据，由于其纪录的是出租车的经纬度位置，而非地点。因此，我们需要首先将出租车出现的区域离散化^③，将GPS经纬度位置对应到相应的区域。

这里我们用节点的权值以及各个节点的访问时间的分别来反映轨迹的空间和时间特征^④。具体来说，对于某个节点 v ，我们赋予 v 两个属性：权值(用 $w(v)$ 表示) 和访问该节点的时间分布，其中节点的权值等于这个节点对应的地点在这条轨迹中出现次数。

① 引用自维基百科: http://en.wikipedia.org/wiki/Bhattacharyya_distance

② 在此文中，如果没有特别说明，随机变量指离散随机变量

③ 我们通过用经纬度将区域分为长方形小格将区域离散化

④ 目前的相似度度量方法是最初的比较基础的版本，仅仅考虑了这两个特征。在下一个版本中，我们将会多增加表示轨迹在 k 个地点转移的多元组，来刻画轨迹局部的转移特征

2.3.2.2 行为网络相似度比较

将轨迹映射到行为网络后，我们就可以通过比较行为网络属性的差异进而来确定对应的轨迹的相似度。具体来说，我们将每个节点对应的地点在对应的轨迹中出现的次数作为这个节点的权值，通过比较两个网络的节点的权值的分布^① 来衡量这两个轨迹的空间相似度。除此之外，我们对每个节点建立它的访问时间的分布^②，通过比较两个网络中对应节点的访问时间的分布，我们可以得到两个网络的时间相似度。分别如下：

1. 轨迹空间相似度 对应第 i 条轨迹 $T^{(i)}$ ，构建它的行为网络 $N^{(i)}$ ，对于 $N^{(i)}$ 中的第 j 个节点 $v_j^{(i)}$ ，我们根据 $v_j^{(i)}$ 在轨迹中出现的次数赋予其权值 $w(v_j^{(i)})$ 。将轨迹 $T^{(i)}$ 出现的地点(将 $T^{(i)}$ 出现的地点的集合记为 $\mathcal{L}^{(i)}$) 看作(离散) 随机变量 $X_{T^{(i)}}$ ， $X_{T^{(i)}}$ 出现在第 j 个地点的概率 $p_{X_{T^{(i)}}}(j) = w(v_j^{(i)}) / \sum_j w(v_j^{(i)})$ ，则第 m 条轨迹 $T^{(m)}$ 和第 n 条轨迹 $T^{(n)}$ 的空间距离 $Distance_{space}(m, n)$ 可由2-5 得到：

$$Distance_{space}(m, n) = D_B(p_{X_{T^{(m)}}}, p_{X_{T^{(n)}}})$$

$$where : BC(p_{X_{T^{(m)}}}, p_{X_{T^{(n)}}}) = \sum_{j \in \mathcal{L}^{(i)}} \sqrt{p_{X_{T^{(m)}}}(j) \cdot p_{X_{T^{(n)}}}(j)} \quad (2-6)$$

2. 轨迹时间相似度 对于轨迹 $T^{(i)}$ 中的第 j 个节点 $v_j^{(i)}$ ，则访问这个节点的时间也可以看作随机变量，用 $Y_{v_j^{(i)}}$ 表示，简记为 $Y_j^{(i)}$ 。将 $Y_j^{(i)}$ 离散化后^③ (将所有时间段的集合记为 \mathcal{P})，假设 $Y_j^{(i)}$ 出现在第 k 时间段内的次数为 $c(Y_{j|k}^{(i)})$ 则 $Y_j^{(i)}$ 出现在第 k 时间段内的概率 $q_{Y_j^{(i)}}(k) = c(Y_{j|k}^{(i)}) / \sum_j c(Y_{j|k}^{(i)})$ ，第 m 条轨迹 $T^{(m)}$ 和第 n 条轨迹 $T^{(n)}$ 的时间距离 $Distance_{time}(m, n)$ 可表示为：

$$Distance_{time}(m, n) = \frac{1}{N} \sum_{j=1}^N D_B(q_{Y_j^{(m)}}, q_{Y_j^{(n)}})$$

$$where : BC(q_{Y_j^{(m)}}, q_{Y_j^{(n)}}) = \sum_{k \in \mathcal{P}} \sqrt{q_{Y_j^{(m)}}(k) \cdot q_{Y_j^{(n)}}(k)} \quad (2-7)$$

轨迹 $T^{(m)}$ 和轨迹 $T^{(n)}$ 的距离 $Distance(m, n)$ 最终由上述两部分距离共同决定，即：

① 将出现的地点作为离散随机变量，地点出现的次数(归一化后) 可以看作离散随机变量的概率分布
 ② 对于每个节点来说，访问它的时间可以看作随机变量，每个时间段出现的次数(归一化后) 可以看作访问时间的概率分布
 ③ 我们以10 分钟为时间单位对 $Y_j^{(i)}$ 进行离散化处理

$$Distance(m, n) = Distance_{space} \times Distance_{time} \quad (2-8)$$

而轨迹 $T^{(m)}$ 和轨迹 $T^{(n)}$ 的相似度 $S(m, n)$ 可以简单地由 $Distance(m, n)$ 得到:

$$S(m, n) = \frac{1}{Distance(m, n)} \quad (2-9)$$

2.4 实验

在这部分中, 我们在北京市24辆出租车^{①[23]}的轨迹上测试了2.3节提出的轨迹相似度算法, 即先将这24条轨迹用2.3.2.1节中的方法映射到行为网络^②, 之后根据公式(2-6) (2-7) (2-8) (2-9) 计算这些出租车轨迹的相似度, 结果见附录B.1。

附录B.1 包含三张表格, 分别为这24辆出租车轨迹的空间不相似度(附录B-1)、时间不相似度(附录B-2) 以及最终的不相似度(附录B-3). 下面, 我们随机选择一辆出租车, 分析一下与它距离最远(最不相似) 以及距离最近(最相似) 的出租车的轨迹模式。

例如我们观察第1辆出租车(T-Drive中编号为366), 它和另外23辆出租车的距离分别为: 11.6, 11.1, 19.4, 16.4, 13.9, 11.7, 14.0, 10.9, 18.6, 12.6, 27.6, 21.9, 21.3, **6.1**, 24.1, 11.5, 14.5, 18.5, 18.5, 10.4, 7.9, 12.6, **38.8**. 与第一辆出租车行为模式最不相近的是第24辆出租车(编号为9754), 最相似的为第15辆车(编号为6275)。第一辆出租车和其它出租车的空间距离(公式2-6) 分别为: 1.6, 1.4, 2.5, 1.9, 1.8, 1.4, 1.8, 1.4, 2.2, 1.6, 3.3, 2.8, 2.5, 0.9, 2.7, 1.5, 1.7, 2.1, 2.2, 1.5, 1.2, **0.6**, **4.1**. 其中与第15辆出租车的距离为0.9, 仅大于和第23辆出租车0.6的距离; 它与第24辆出租车的空间距离为4.1, 大于与其它车辆的距离。下面我们就来分析一下第1辆车与第15辆车、第24辆车的轨迹特征。

这里, 我们用GPSPrune可视化了第1辆、第15辆、第24辆出租车的轨迹, 我们可以发现, 第1辆车与第15辆车的轨迹有许多重合的区域, 而与第24辆车的轨迹重合的区域很少, 因此与第15辆车的空间距离较小(相似度高), 与第24辆车的空间距离较大(相似度低)。

① 这24辆出租车的轨迹来源于公开数据集T-Drive, 第4章有对这个数据集详细的说明

② 划分区域时, 经度范围: 115.25-117.30, 纬度范围: 39.27-41.03, 经度间隔0.02, 纬度间隔0.02

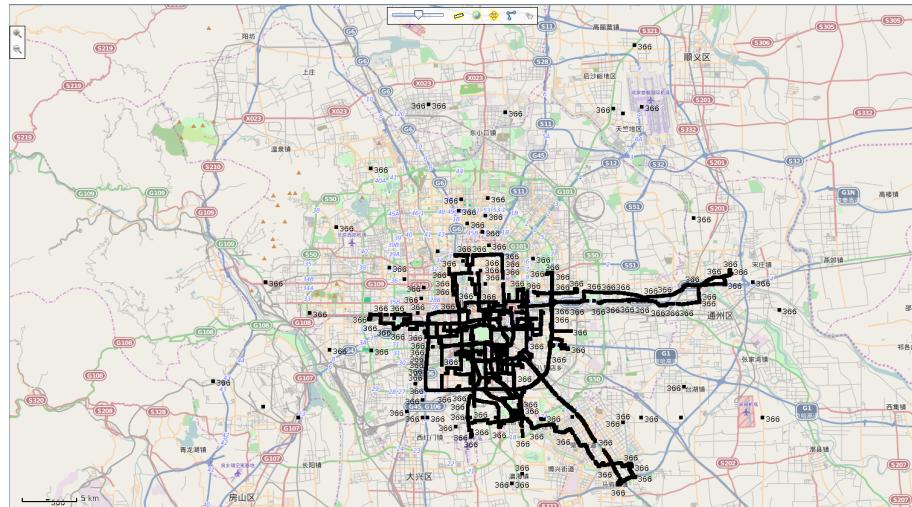


图 2-3 出租车1 的轨迹

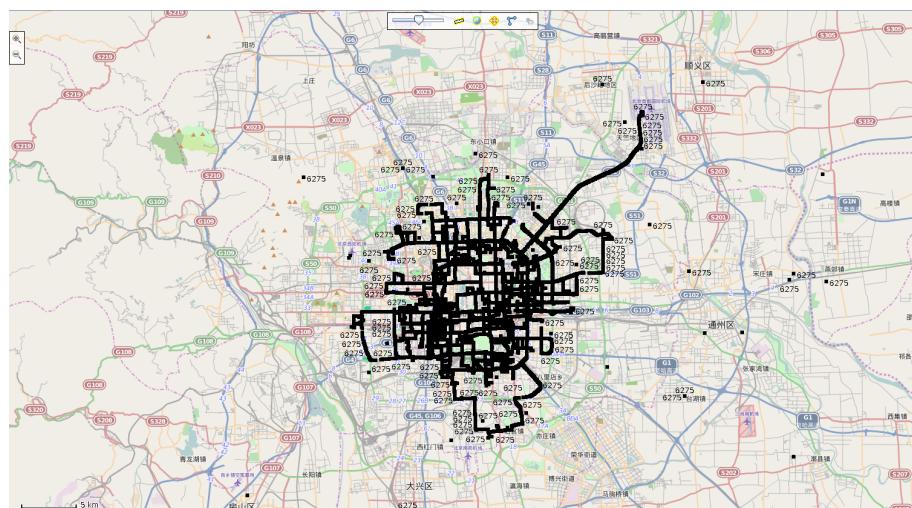


图 2-4 出租车15 的轨迹

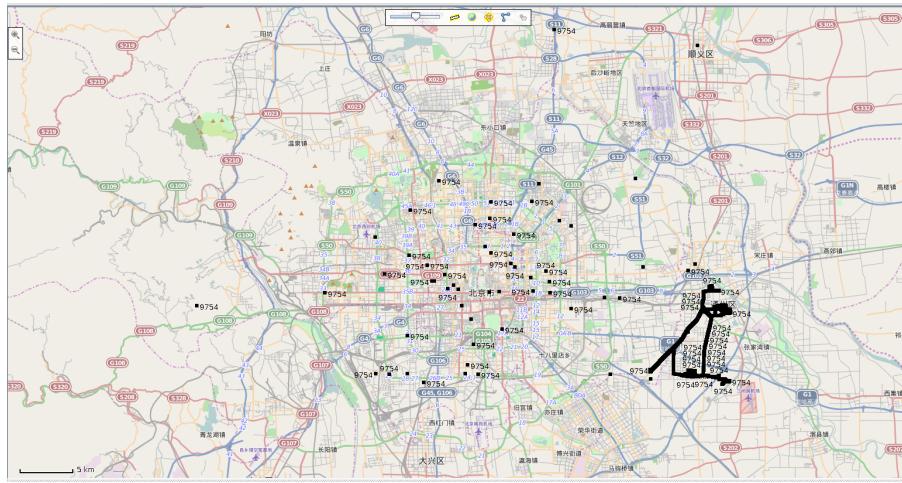


图 2-5 出租车24 的轨迹

对于第24辆车来说，它和其它每辆车的距离都比较大，见附录B-1, B-2, B-3，这是由于第24辆车的主要活动区域并不在北京市，而其它车辆主要在北京市活动。

2.5 总结

在这篇文章中，我们从空间信息和时间信息两个角度出发，考虑的轨迹的相似度度量。具体来说，我们首先将一个轨迹映射到一个行为网络上，并赋予行为网络一些属性使得它可以反映轨迹的时间特征和空间特征；之后，通过比较行为网络的反映了轨迹信息的属性来衡量轨迹之间的相似度。通过在出租车轨迹上的实验，进一步说明了上述方法可以放映出租车轨迹之间的相似程度。在第4章中，我们将以这个相似度指标为基础，完成这些出租车的聚类。

第3章 基于动态交互的社团挖掘算法

聚类分析是数据挖掘领域一项非常重要的方法，在许多领域都有重要应用^[24]。正如前面所说，聚类分析的任务是将相似的对象分为一类，不相似的对象分到不同类。因此聚类分析通常需要首先定义对象与对象之间的相似度，在此基础上，应用聚类算法将相似的对象分为一类，不相似的对象分到不同类。聚类算法可以分为两类，第一类是应用于向量空间上的聚类算法，例如DBSCAN, OPTICS 等等，这类算法的研究对象用 n 维向量表示。另一类是应用于测度空间上的聚类算法，例如 k -means, 谱聚类等等，当提供对象与对象之间的相似度后，这类算法即可将研究对象分为不同的类。当然，第二类算法也完全可以应用于向量空间，因为在向量空间上，我们很容易定义向量与向量之间的相似度。此外，基于测度空间的聚类与图聚类非常相似，在许多情况下，这两者可以相互转化^①。

对于图聚类(社团挖掘)这个问题，目前已经有许多经典的算法被提出来，例如：Ncut 算法, Louvain 算法, 等等。但是许多经典的社团挖掘(图聚类)算法存在一些问题，比如广泛应用的基于模体(modularity) 社团挖掘算法假设网络符合零模型，而这个假设在实际的特别是大网络中并不合理；另外一个经典的Ncut^[25] 算法通常得到的是规模相近社团^②，而现实中社团的规模可能相差很大^[26]。除此之外，受复杂度的限制，许多经典的算法并不能有效地处理现实中出现的大规模网络。因此有效地挖掘实际存在的大规模网络中的社团结构依然是一个挑战。

在这篇文章中，我们提出了一个新的社团挖掘算法: *Attractor*。不同于现有的许多算法来优化某个人为指定的指标，例如modularity, cut, normalized cut, 我们是将网络看作一个动态系统，通过节点与节点之间的交互，自动地发现社团结构。我们将会看到Attractor 可以有效地检测网络中存在的社团结构，并且不受社团规模的限制，因此可以较好地发现小规模的社团或是奇异点；此外，由于网络中各个节点只与周围节点交互，因此Attractor 的复杂度很低($O(|E|)$)，因此Attractor 可以用于大规模网络。

① 给定一个阈值，我们可以将相似矩阵转化为一个网络；反之，通过定义相似度，即可将网络转化为相似矩阵

② 社团规模指的是这个社团中节点的个数

这一章剩余内容的结构安排如下：3.1 将会介绍我们算法的基本思想以及*Attractor* 算法的主要贡献；3.2 简单地回顾了现有的相关社团挖掘(图聚类)算法；3.3 将会具体地介绍*Attractor* 算法；3.4 是在人工数据集以及真实数据集上实验；最后3.5 是对这个工作的总结。这章介绍的关于*Attractor* 的工作已经被*SIGKDD'2015*^① 会议接受。

3.1 基本思想与主要贡献

3.1.1 基本思想

从社会学角度来讲，社团可以看作一群具有相对紧密联系的个体^[27]。因此我们想知道是否可以通过模拟人与人之间的交互(联系)来挖掘社团结构，即我们希望人与人交互过程中，同一个社团中的人的联系变得更加紧密，进而聚集到一起。具体来说，我们是将网络看作一个节点与节点交互的动态系统，但是与许多传统方法不同的是，我们并不考虑节点状态的变化，而是考察节点与节点之间距离的变化(网络中边的距离的变化)。在交互过程中(见3.3)，开始时每条边有一个初始的距离；之后节点与节点开始交互，在此过程中，同一社团的节点之间的距离会变短，属于不同社团的节点之间的距离会增加；最终达到一个稳定的状态(称这个稳定的状态为*Attractor*)，于是我们可以通过移除距离为1的边来自动地得到社团结构。

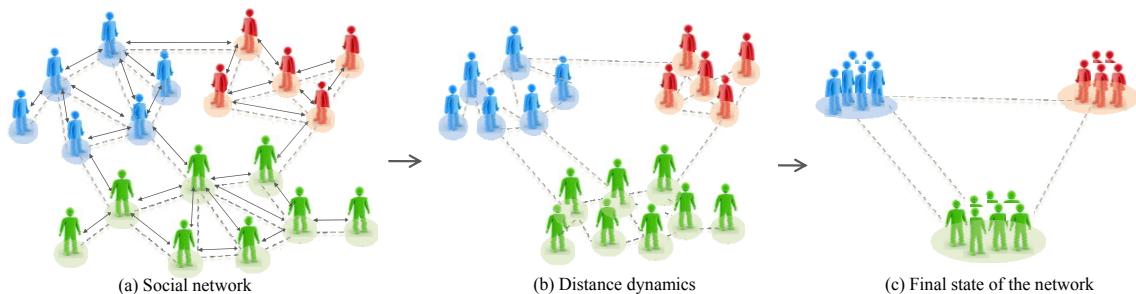


图 3-1 基本思想

3.1.2 主要贡献

通过节点之间的交互过程，*Attractor* 算法可以有效地发现实际网络中存在的社团结构，特别的，*Attractor* 有以下主要贡献：

^① 21st ACM SIGKDD conference, 10 - 13 August 2015, Sydney, Australia

- 直观的社团挖掘算法:** 不同于许多经典算法来优化某个人为定义的指标, *Attractor* 从一个新的(动态交互) 角度来探索社团结构。通过三种交互模式(见3.3), *Attractor* 算法可以直观地并且有效地发现社团结构。
- 小社团与奇异点检测:** 由于节点之间的距离变化由节点所处的网络的拓扑结构确定, 因此*Attractor* 算法可以有效地检测任意规模的社团以及奇异点。
- 可扩展性:** 由于每个节点只与周围节点交互, 因此*Attractor* 算法具有较低的时间复杂度($O(|E|)$), 这使得*Attractor* 算法可以处理大规模网络。并且由于节点交互时相互独立, 因此*Attractor* 非常适合用并行算法实现(甚至可以将整个网络分为几个部分用不同的计算机上分别进行处理)。

3.2 相关工作回顾

正如前文所提到, 近些年, 许多经典的社团挖掘算法被相继提出, 例如*Cut*, *Ncut*, *Modularity*, *MCL*, *Metis* 等等, 由于篇幅限制, 这里仅仅简单地回顾了与本文最相关的几个算法。关于社团挖掘算法的详细的介绍, 请见综述文献^[24] 和^[28]。

基于某个指标删边的社团挖掘方法. 这类方法以*Ncut* 和*Modularity* 为代表, 它们首先对每条边赋予一个权值(例如*Ncut* 或者*Modularity*), 进而删去一些边使得在这种指标下, 整个网络达到最优。*Cut*^[29] 是其中一个比较出名的工作. 在^[29]中, 作者将通过移除一些边将一个网络分成 k 个社团, 其中移除的这些边尽可能地少。为了克服*Cut* 算法常常分出过多的少社团这个问题, Shi 等人把社团规模考虑了进来, 提出了非常出名的*Normalized cut* 指标, *Ncut* 算法通过优化*Normalized cut* 指标来找到网络中存在的社团^[25]。在实际中, 常常使用特征值分解^[30] 的方法来加速计算, 在许多情况下, 使用谱分解进行聚类的方法也被称为谱方法。此外, 还有一类基于*Modularity* 指标的社团挖掘方法也非常流行^{[31][32]}, 为了减小时间开支, 贪婪算法^[33]以及模拟退火^[34]等方法被加入来减少算法运行时间。

可扩展的大规模网络处理方法 为了处理现实中存在的大规模网络, 许多算法被相继提出^[35] ^[36] ^[37]. 例如*Metis* 是一类可以用于大规模网络的层次社团挖掘算法, 它每次将原来的社团分成两个部分, 逐渐得到越来越精细的社团结构。另外一种在生物领域应用广泛的聚类算法*MCL*^[37], 通过模拟网络中的随机流来分隔网络。

3.3 基于动态交互的社团挖掘算法

3.3.1 准备工作

在介绍我们的社团挖掘算法之前，首先来约定一些最基本的规定。

定义一 (无向图) 用 $G = (V, E, W)$ 来表示无向图，其中 V 是节点构成的集合， E 是边构成的集合， W 是边的权重构成的集合。 $e = \{u, v\} \in E$ 表示节点 u 和 v 节点之间存在一条连边， $w(u, v)$ 表示 $\{u, v\} \in E$ 的权重。对于 $\forall e = \{u, v\} \in E, w(u, v) = 1$ ，则 G 是一个无向无权图。

定义二 (节点的邻居) 在无向图 $G = (V, E, W)$ 中， u 节点的邻居 $\Gamma(u)$ 包含节点 u 以及与节点 u 有边相连的其它节点，

$$\Gamma(u) = \{v \in V | \{u, v\} \in E\} \cup \{u\}$$

进一步，我们使用 *Jaccard Distance*^[38] 来度量初始时节点与节点之间距离。

定义三 (JACCARD 距离) 在无向图 $G = (V, E, W)$ 中， u 节点和 v 节点之间的 Jaccard 距离定义如下：

$$d(u, v) = 1 - \frac{|\Gamma(u) \cap \Gamma(v)|}{|\Gamma(u) \cup \Gamma(v)|}$$

对于有权图，节点 u 和节点 v 之间的 Jaccard 距离定义如下：

$$d(u, v) = 1 - \frac{\sum_{x \in \Gamma(u) \cap \Gamma(v)} (w(u, x) + w(v, x))}{\sum_{\{x, y\} \in E; x, y \in \Gamma(u) \cup \Gamma(v)} w(x, y)}$$

3.3.2 交互模型

为了研究节点与节点之间的距离在交互过程中的变化，我们首先需要定义交互模型，包括交互范围以及交互模式。

交互范围 网络中的边非常直观地显示了各个节点的交互范围，因为我们可以很自然地假设：假如两个节点之间有边相连，那么这两个节点之间存在交互作用。

交互模式 在确定交互范围后，另外一个关键的问题就是确定交互模式。由于社团结构由节点与节点之间最终的距离(边的距离)确定，因此我们考察边的距离的变化。假设 $e = \{u, v\} \in E$ 是连接两个相邻节点 u 和 v 的一条边， $d(u, v)$ 是 $e = \{u, v\}$ 的初始时的距离，我们考察 e 的距离是怎么变化的。显然，

边 $e = \{u, v\}$ 的距离的变化是由它连接的两个节点 u 和 v 的变化引起的，而与 u 和 v 有边相邻的节点(u 和 v 的邻居)都会对 u 和 v 产生作用。根据与 u 和 v 有边相邻的节点拓扑位置的不同，我们可以将 u 和 v 的邻居分成三类，对应于 $e = \{u, v\}$ 发生变化的三种情景。

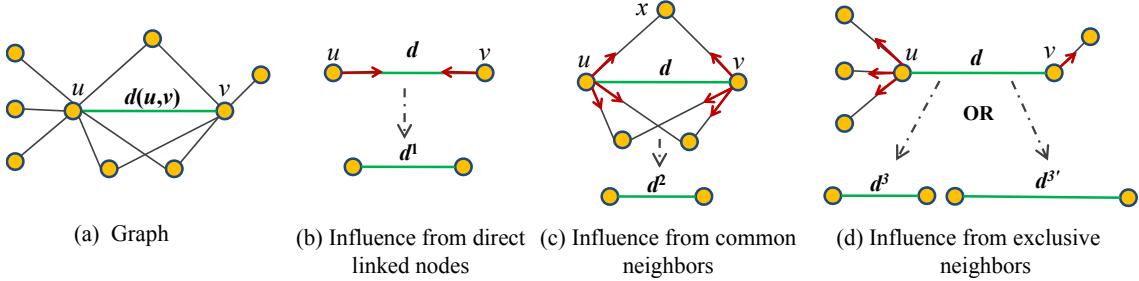


图 3-2 三种交互模式示意图

PATTERN 1: 来自 u 和 v 的影响。在交互过程中，假如两个节点 u 和 v 之间有边相连，那么我们假设其中一个节点吸引着另外一个节点靠近自己，因此这两个节点之间的相互作用使得它们的距离倾向于变小(相似度增加)。正如社会网络中，每个人通过分享自己的兴趣爱好给他的朋友，使得他的朋友与自己的紧密程度增加。假设用 DI 来表示 u 和 v 对 $e = \{u, v\}$ 这条边距离 $d(u, v)$ 变化的直接影响， DI 定义如下：

$$DI = - \left(\frac{f(1 - d(u, v))}{deg(u)} + \frac{f(1 - d(u, v))}{deg(v)} \right) \quad (3-1)$$

其中 $deg(u)$ 是节点 u 的度， $f(\cdot)$ 是耦合函数(我们选用 $\sin(\cdot)$ 函数作为耦合函数)。 $1 - d(u, v)$ 等于节点 u 和节点 v 之间的紧密程度(相似度)，若节点 u 和节点 v 的距离越近(越紧密)，则它们之间的作用越强。 $\frac{1}{deg(u)}$ 考虑了节点的度的影响，一般来说，当一个节点的度越大时，那么这个节点可以从更多的途径获得知识，因此这个节点越不容易被某一个途径来的影响；相反，假如一个节点的度很小，也就是说它只有非常有限的途径来获得信息，那么某一个途径来的影响则容易对它造成影响。

PATTERN 2: 来自 u 和 v 共同邻居的影响 这里我们考虑另外一类对 $e = \{u, v\}$ 这条边的距离产生影响的节点：节点 u 和节点 v 的共同邻居。假设节点 x 属于节点 u 和节点 v 的共同邻居构成的集合 $CN = (\Gamma(u) - u) \cap (\Gamma(v) - v)$ ，由于节点 x 与节点 u 和节点 v 都有边相连，因此节点 x 将节点 u 和节点 v 都拉向自己，从而间接地减小节点 u 和节点 v 的距离。假设用 CI 表示节点 u 和节点 v 共同邻居对 $e = \{u, v\}$ 这条

边距离的影响, CI 定义如下:

$$CI = - \sum_{x \in CN} \left(\frac{1}{deg(u)} \cdot f(1 - d(x, u)) \cdot (1 - d(x, v)) + \frac{1}{deg(v)} \cdot f(1 - d(x, v)) \cdot (1 - d(x, u)) \right) \quad (3-2)$$

与3-1 相比, 有额外两项($1 - d(x, v)$) 和($1 - d(x, u)$) 被加入到3-2 中来量化共同邻居 x 对 $e = \{u, v\}$ 这条边距离的影响。例如, 我们考察 x 通过 u 对 $e = \{u, v\}$ 的作用时, 假如 x 和 v 越相似, 那么 x 和 v 对 $e = \{u, v\}$ 的作用就越相似, 而($1 - d(x, v)$) 和($1 - d(x, u)$) 分别表示了 x 和 v 的相似度, 以及 x 和 u 的相似度。当 x 和 v 的相似度为 1 时(x 和 v 可以看作完全一样的节点), 那么 x 对 $d(u, v)$ 的影响可以转化为第一种(3-1) 交互模式。

PATTERN 3: 来自 u 和 v 单独邻居的影响 除了上述两类节点会对 $e = \{u, v\}$ 这条边的距离产生影响外, 还有 u 的单独邻居 $EN(u) = \Gamma(u) - \Gamma(u) \cap \Gamma(v)$ 以及 v 的单独的邻居 $EN(v) = \Gamma(v) - \Gamma(u) \cap \Gamma(v)$ 会对 $e = \{u, v\}$ 的距离产生影响。与前两种交互模式类似, 节点 u 的单独的邻居会将 u 拉向自己(节点 v 的单独的邻居会将 v 拉向自己), 但是我们并不知道 u 和 v 到底会变近还是变远, 因为假如节点 u 的单独的邻居离节点 v 较远的话, u 靠近它单独的邻居的结果是 u 和 v 的距离增加; 反之假如节点 u 的单独的邻居离节点 v 较近的话, u 靠近它单独的邻居的结果是 u 和 v 的距离减小(v 单独的邻居的作用效果类似)。因此为了判断 u 和 v 各自的单独的邻居 $EN(u)$ 和 $EN(v)$ 对 $d(u, v)$ 的影响, 我们需要给定一个相似度的阈值用来判断 u 的单独的邻居与 v 是否相似, 以及 v 的单独的邻居与 u 是否相似。如果 u 的单独的邻居与 v 相似的话(距离小), u 与它单独邻居之间的距离变小的结果是 u 和 v 之间的距离也变小; 反之如果 u 的单独的邻居与 v 不相似的话(距离大), u 与它单独邻居之间的距离变小的结果是 u 和 v 之间的距离变大。这里引入一个紧密度指标 λ 来刻画 $EN(u)$ 和 $EN(v)$ 对 $d(u, v)$ 的影响 EI 。

$$EI = - \sum_{x \in EN(u)} \left(\frac{1}{deg(u)} \cdot f(1 - d(x, u)) \cdot \rho(x, u) \right) - \sum_{y \in EN(v)} \left(\frac{1}{deg(v)} \cdot f(1 - d(y, v)) \cdot \rho(y, v) \right) \quad (3-3)$$

其中

$$\rho(x, u) = \begin{cases} (1 - d(x, v)) & (1 - d(x, v)) \geq \lambda \\ (1 - d(x, v)) - \lambda & \text{otherwise} \end{cases} \quad (3-4)$$

最终，节点 u 和节点 v 在 $t+1$ 时刻的距离 $d(u, v, t + 1)$ 可以由 t 时刻 u 和 v 的距离 $d(u, v, t)$ 以及 $DI(t)$, $CI(t)$ and $EI(t)$ 确定：

$$d(u, v, t + 1) = d(u, v, t) + DI(t) + CI(t) + EI(t) \quad (3-5)$$

3.3.3 Attractor 算法

3.3.3.1 Attractor 算法

基于上述交互模型，下面介绍我们的*Attractor* 算法。

1. 开始时($t = 0$), 每条边被赋予一个权重。这里我们使用的是Jaccard 距离(见3.3.1 和3.3.1)。
2. 之后随着时间的演化，每个节点与其周围的节点进行交互，每条边的距离的变化由交互模型中的三种不同交互方式确定：(Eq. (3-1), Eq. (3-2) and Eq. (3-3)). 在交互过程中，同一个社团内的节点将会相互靠近，属于不同社团的节点之间的距离将会增加。
3. 最终，当所有的边的距离不再变化时，我们将边长为1的边去掉，之后依然有边相连的节点即可看作一个社团。

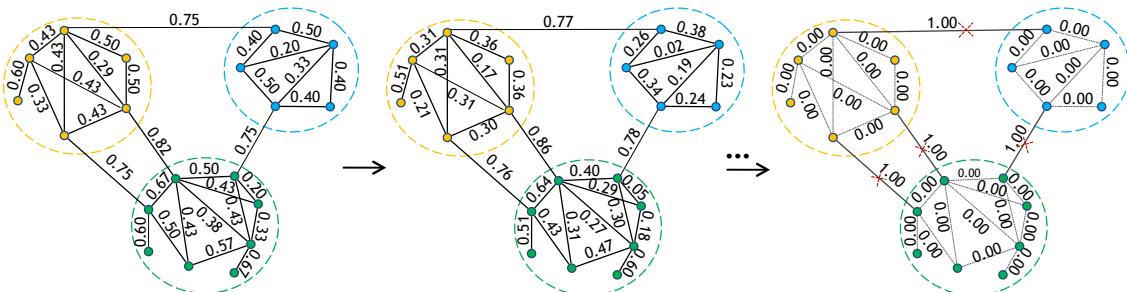


图 3-3 动态交互过程示意图. (1) 初始时的网络; (2) 经历一个时间步后的网络; (3) 最终稳定时的网络

图3-3 是一个模拟网络动态交互过程的示意图，3-3(a)-(c) 表示了网络交互时的三种状态($t = 0$ 到 $t = 9$)。开始时(3-3(a), $t = 0$), 网络的每条边有一个初始长 度；之后节点开始交互，边的距离发生变化，经过一个时间步后($t = 1$), 边的距离

如3-3(b)表示；最后经过9个时间步的交互更新后，整个网络的距离不再发生变化，将边长为1的边去掉后即可得到最终的社团结构。*Attractor* 算法的伪代码见附录A.1。

3.3.3.2 对耦合参数 λ 的讨论

正如上述讨论，对于某条边来说，耦合度参数确定了它连接的节点的单独的邻居对这条边的影响(见3-4). 一般而言， λ 越大，则可以得到更多的精细的社团； λ 越小，那么得到的社团数量较少，社团规模较大。通过设置不同的 λ ，*Attractor* 算法可以从不同的尺度上检测网络的社团结构。更重要的是，相比较于许多需要指定划分的社团的个数的算法， λ 的设置非常直观并且容易。图3-4 展示了当 λ 取 $[0, 1]$ 时的人工网络的结果。我们可以发现 λ 在一个较大范围取值时($0.2 - 0.7$)，*Attractor* 算法都可以得到理想的结果，并且当 λ 从小到大变化时，*Attractor* 算法可以在不同层次上揭示网络的社团结构。

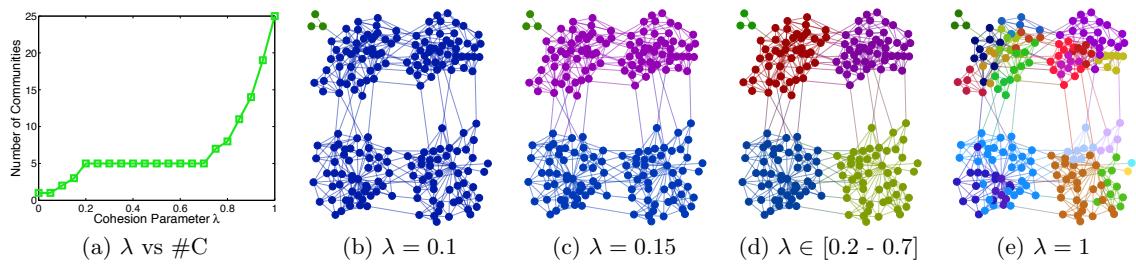


图 3-4 耦合参数 λ 取值与聚类结果关系示意图

3.3.4 时间复杂度分析

这里我们来分析*Attractor* 算法的时间复杂度。

最开始时，我们需要计算每条边的长度，时间开销为 $O(|E|)$ ，之后，对于每条边 $e = u, v$ 来说，每次交互时，需要计算 u 和 v 单独的邻居的距离，因此时间开销为 $O(k \cdot |E|)$ ，假设交互 T 个时间步，那么总共的时间开销为 $O(|E| + T \cdot (k \cdot |E|))$ 。因此*Attractor* 算法的时间开销依然是 $O(|E|)$ ，也就是与网络的边数成正比。

3.4 实验

为了说明*Attractor* 算法的优势，我们在人工数据集以及真实数据集上测试了*Attractor*，并且与一系列经典的算法进行了比较。

比较算法 用于比较的经典算法有：*Ncut*^[25], *Modularity*^[32], *Metis*^[35], *MCL*^[37], *Louvain*^[33], *Infomap*^[34].

我们将 $\lambda = 0.5$ 设置为*Attractor* 的默认参数。实验是在3.4 GHz CPU 和32.0 GB RAM的电脑上完成的。

评价指标 为了比较*Attractor* 算法与这些经典算法的结果，我们选用了目前最常用的几个指标。

- 对于社团结构已知的网络， 由于网络已有标签， 因此可以通过比较算法得到的结果与真实结果之间的差异来评价算法的好坏。这里， 我们采用了三个最常用的指标：*Normalized Mutual Information (NMI)*^[39], *Adjusted Rand Index (ARI)*^[40] 和*Cluster Purity*^①.
- 对于社团结构未知的网络， 由于网络的真实社团结构未知， 因此在这些网络上来比较各个算法的结果不容易。在这里， 我们选了2个被广泛使用的指标*modularity*^[31] 和*normalized cut (ncut)*^[25] 来评价各个算法。需要注意的是， 一些算法(例如基于*modularity* 和基于*ncut* 的算法) 正是来优化的这两个指标， 因此这种比较方法并不公平。

3.4.1 人工数据集上的对比试验

首先， 我们构建了有真实社团标签的人工数据集来比较*Attractor* 与其它经典算法。为了更好地模拟现实中的网络以及尽可能公平， 我们选用LFR 网络^[41]. 在LFR 网络中， 节点与属于其它社团内的节点连边占它所有边的比例被定义为混合参数(*mixing parameter*), 记作 μ . 当 μ 越小时， 正确划分社团越容易， 当 μ 越大时， 正确划分社团越困难。

① 见<http://nlp.stanford.edu/IR-book/html/htmledition/evaluation-of-clustering-1.html>

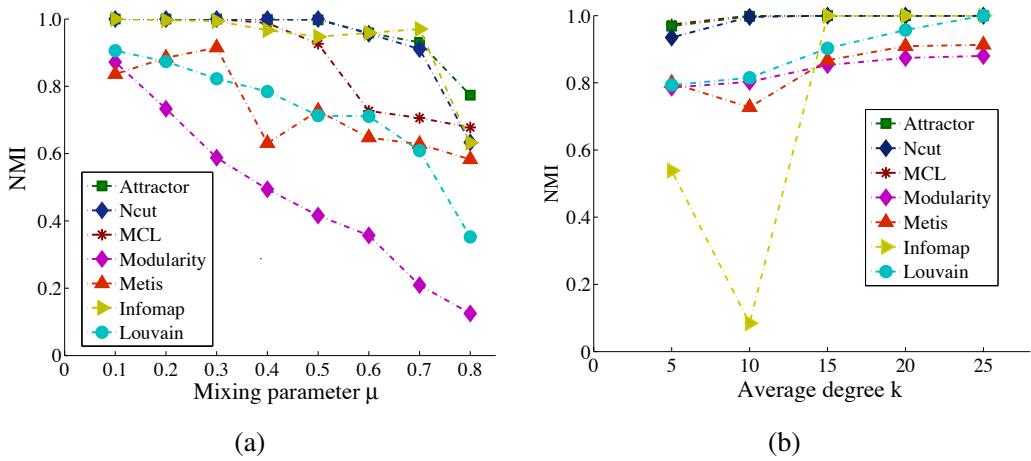


图 3-5 人工数据集实验结果

(a)LFR1; (b)LFR2

3.4.1.1 改变噪声边比例

首先我们比较这些算法在网络具有不同噪声边(社团与社团之间的边的比例)时的表现(结果见3-5(a))。通过调节LRF 网络中的 μ 参数(μ 取0.1 到0.8)，我们即可得到具有不同噪声边比例的网络。所有的网络都有2000个节点，以及它们的平均度 $k = 20$.

从3-5(a) 中可以发现，*Attractor*, *Ncut*, *MCL* 和*Infomap* 在 $\mu = 40$ (每个节点有40% 的边与其它社团中的节点相连)时依然可以得到很好的结果，它们的结果随着噪声边的增加而变差，但是相比之下，*Attractor* 对噪声更不敏感。*Modularity* 和*Louvain* (基于*modularity* 指标) 两种算法相比上述4种算法明显要差一些。至于*Metis*, 它的表现十分波动，并且当 μ 超过40% 时，*Metis* 得到的社团的质量在明显下降。

3.4.1.2 改变网络稀疏度

之后，我们比较网络的稀疏程度对这些算法的影响。这里，LRF 网络中的 μ 参数被固定为0.1 (网络都有2000个节点)，我们改变网络的平均度 k ，结果见3-5(a).

结果显示*Attractor*, *MCL*, *Ncut* 在这些网络(平均度 k 从25取到5) 都有不错的表现，即使在平均度 $k = 5$ 的稀疏网络上，这三个算法依然可以有效地找到社团结构，并且*Attractor* 的结果要更好一点。*Louvain*, *Metis* 和*Modularity* 对网络的稀疏程度要敏感一点。对于*Infomap* 算法，当网络的平均度为10时，它的结果是一个奇异点。

3.4.2 真实数据集上的对比试验

这部分，我们在真实的数据集上比较这些算法。这些数据集都是公开的，在`https://networkdata.ics.uci.edu/index.php` 或者`http://snap.stanford.edu/data/` 可以下载。这些网络的特征见表3-1。

表 3-1 真实网络统计特征(AD: average degree; CC: cluster coefficient)

Data Sets	V	E	#Class	AD	CC
Zarachy	34	78	2	4.588	0.571
Football	115	613	11	10.66	0.403
Polbooks	105	441	3	8.400	0.488
Amazon	334863	925872	151037	5.530	0.397
Collaboration	9875	25973	-	5.260	0.312
Friendship	58228	214078	-	7.353	0.172
Road	1088092	1541898	-	2.834	0.047

3.4.2.1 社团结构已知的网络

首先，我们在四个已知社团结构的网络上比较这些算法，选用的评价指标是：*NMI*, *ARI* 和 *purity*. 结果见表3-2.

表 3-2 有标签网络上各算法比较

	Zarachy			Football			Polbooks			Amazon		
	NMI	ARI	Pur.									
Attractor	0.859	0.939	1.000	0.923	0.897	0.930	0.559	0.680	0.857	0.931	0.580	0.998
Ncut	0.833	0.882	0.970	0.923	0.897	0.930	0.534	0.645	0.829	-	-	-
Modularity	0.577	0.680	0.970	0.596	0.474	0.574	0.508	0.638	0.838	-	-	-
Metis	0.836	0.882	0.970	0.393	0.095	0.339	0.502	0.516	0.781	0.761	0.092	0.989
MCL	0.833	0.882	0.970	0.923	0.897	0.930	0.455	0.594	0.857	0.902	0.490	0.991
Louvain	0.524	0.541	1.000	0.858	0.807	0.870	0.440	0.537	0.857	0.738	0.384	0.384
Infomap	0.593	0.702	0.971	0.906	0.857	0.904	0.476	0.646	0.848	0.209	0.009	0.077

Zachary's karate club network: ^[42] Zachary 空手道俱乐部网络是一个非常有名的小网络，它反映了一个空手道俱乐部成员之间的朋友关系。并且我们已经知道，这个空手道网络中的成员可以分为2个社团^①。图3-6 显示Attractor 算法有效的监测到了这个网络中存在的两个社团，并且在*NMI*, *ARI* 和 *Purity* 这三个指标下来看，Attractor 的结果是最好的。事实上，Attractor 将其中一个节点当作了噪声点，如果我们仔细观察这个点的话，可以发现这个点处于两个社团之间，在实际中，仅仅根据这个网络的拓扑属性很难正确区别这个节点。对于其它算法来

① 这两个社团分别有各自的中心成员：俱乐部经理和教练。经理和教练存在着的分歧正好体现在网络结构中

说, *MCL* 和 *Ncut* 也达到了很好的结果, 但是它们都分错了一个点。从 *NMI*, *ARI* 和 *Purity* 这三个指标来衡量的话, 剩下的算法得到的结果并不理想。

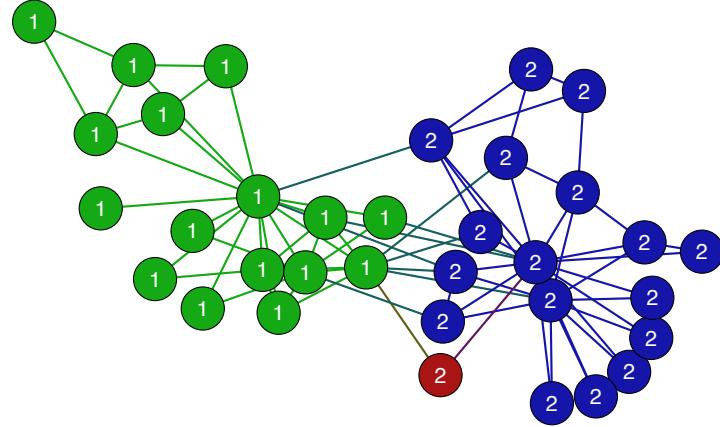


图 3-6 *Attractor* 算法在空手道网络上的结果

橄榄球比赛网络: ^[43] 这个网络包含115个节点, 613条边, 其中节点代表球队, 如果两个节点之间有边相连, 那么这两个球队在2000年秋季赛季中有过比赛。根据这些球队所属协会不同, 它们可以被分成12个社团。图3-7是*Attractor*得到的结果。我们可以发现*Attractor*很好地找到了这些社团($NMI = 0.923$, $ARI = 0.897$, $Purity = 93.0\%$). *MCL* 和 *Ncut* 得到了和*Attractor*相似的结果, 但是*Metis*, *Modularity*, *Louvain* 和 *Infomap* 得到的结果并不好。

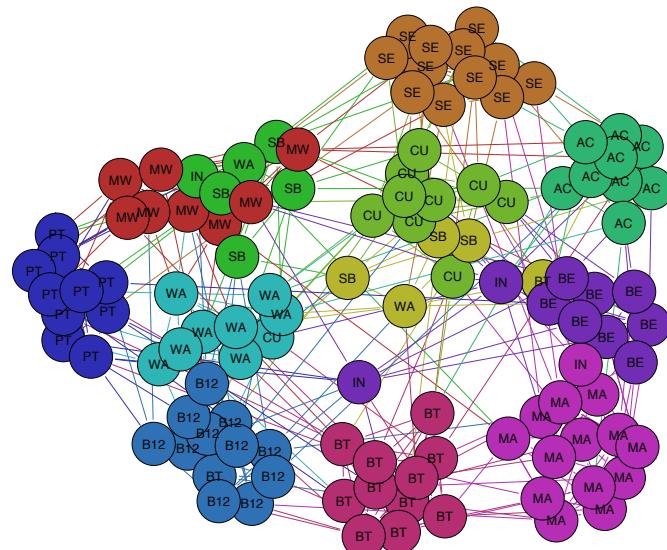


图 3-7 *Attractor* 算法在橄榄球比赛网络上的结果

政治书网络:^① 这个网络中每个节点代表一本政治书，它们的标签“l”，“n”和“c”分别表示它们的观点“激进”，“中立”和“保守”，如果两本书在Amazon.com被同一个读者购买过，那么这两本书之间有一条连边。这些算法相比之下，*Attractor* 得到的结果较好，特别地，激进类和保守类这两个社团结构被较好地检测到(见图3-8)。

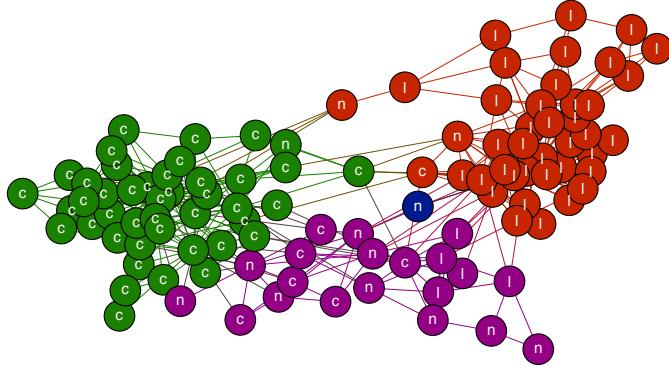


图 3-8 *Attractor* 算法在政治书网络上的结果

亚马逊商品网络: 这个网络包含334,863个节点，925,872条边，其中每个节点根据它的类别被分到某一个社团中。文献^[44] 给出了前5,000个可信度最高的社团。根据这5,000个社团的结果，我们比较了*Attractor* 与其它算法(*Ncut* 和 *Modularity* 这两个算法由于复杂度过高，不能处理这个网络)在这个网络上的结果($NMI = 0.931$, $ARI = 0.580$, $Purity = 0.998$)。*MCL* 也得到了一个较好的结果($NMI = 0.902$)。但是*Metis*, *Louvain* 和 *Infomap* 并不能得到较好的结果。另外，这5个算法得到的社团在 *modularity* 和 *ncut* 这两个指标下的结果见表3-3。

3.4.2.2 社团结构未知的网络

现实中的网络大多规模较大，而社团结构未知。在这些无标签的大规模网络上，由于没有真实标签，因此无法再次使用 NMI , ARI 和 $purity$ 这三个指标对算法进行比较。这里我们选用了另外的两个指标来衡量各个算法得到的结果。另外，由于 *Ncut* 和 *Modularity* 的复杂度很高，使得它们无法处理这些大规模网络，因此只有 *Metis*, *MCL*, *Louvain* 和 *Infomap* 这四种算法与 *Attractor* 进行了比较。结果见表3-3。

科学家合作网络:^[45] Heptb 科学家合作网络由9,875个在高能物理领域发过文章的作者组成，每个节点代表一个作者，假如两个作者是同一篇文章

^① <http://www.orgnet.com/>

表 3-3 Results on Unlabeled Real Networks

	Collaboration			Friendship			Amazon			Road		
	#C	mod.	ncut	#C	mod.	ncut	#C	mod.	ncut	#C	mod.	ncut
Attractor	1384	0.579	1179	8045	0.421	7325	23825	0.741	10811	59919	0.856	25055
Metis	1384	0.309	4217	8045	0.138	53984	23825	0.451	47336	59919	0.673	31542
MCL	2093	0.537	2103	13788	0.319	36723	46557	0.623	47488	86745	0.810	25065
Louvain	475	0.768	10.120	746	0.684	38.340	240	0.926	9.617	492	0.989	2.032
Infomap	456	0.722	5.470	572	0.439	4.104	12	0.4224	0.1249	208	0.660	6.088

的共同作者，那么这两个节点之间有一条边相连。在这个数据集上，假如我们用 *modularity* 和 *ncut* 这两个指标来衡量得到的结果的好坏，*Attractor* 的结果要好于 *Metis* 和 *MCL*，但 *Louvain* 和 *Infomap* 的结果要比 *Attractor* 更好(3-3)。但是需要注意的是，与 *Attractor* 算法相比，*Louvain* 和 *infomap* 得到的社团非常少，从 *modularity* 和 *ncut* 的计算过程可以发现，这样的结果很自然地产生较好的 *modularity* 和 *ncut*。

基于位置信息的朋友关系网络: Brightkite 朋友关系网络包含 8,228 个节点和 214,078 条边。在这个数据集上，*Attractor* 找到了 8,045 个社团并比 *MCL* 和 *Metis* 具有明显优势。不过，和预想的一样，*Louvain* 和 *Infomap* 得到的社团数量很少，这两个算法得到的结果的 *modularity* 和 *ncut* 相比 *Attractor* 更好一些。

道路网络: Pennsylvania 道路网络中节点代表交叉路口或者路的起点或终点，边对应着连接这些地点的道路。这里，由于这个网络非常稀疏，我们将 *Attractor* 的耦合度参数设置为 0.6，*MCL* 的参数设置为 1.4，*Attractor* 找到了 59,919 个社团(*modularity* = 0.856 and *ncut* = 25055)，*MCL* 得到了类似的结果并且比 *Metis* 要好。*Louvain* 和 *Infomap* 找到了很少量的社团。

3.4.3 社团规模分布和噪声点检测

3.4.3.1 社团规模比较

之前已经多次提到，一些经典的算法倾向于将网络分成规模接近的社团^[26]，因此无法有效地检测现实中常常存在的小规模社团，以及当社团规模并不接近时，这些算法的效果并不好。这里，我们选用 Amazon 网络为例，测试了这些算法得到的社团规模分布，并且与 Amazon 网络真实的社团分布进行了比较，结果见图3-9。

从图3-9中可以发现, *Attractor* 可以发现不同规模大小的社团, 特别是许多小规模的社团, 而且分布和真实的社团规模分布最为接近。进一步, 为了证明*Attractor* 找到的这些社团是有意义的, 我们用^[26] 提供的最可信的前5,000个社团作为真实标签, 检测了*Attractor* 算法得到的规模小于30的社团(*Attractor* 找到了1,458个规模小于30的社团), 得到的结果是: $NMI = 0.941$, $ARI = 0.637$ 以及 $Purity = 0.989$, 这说明了*Attractor* 找到的小社团是有意义的。

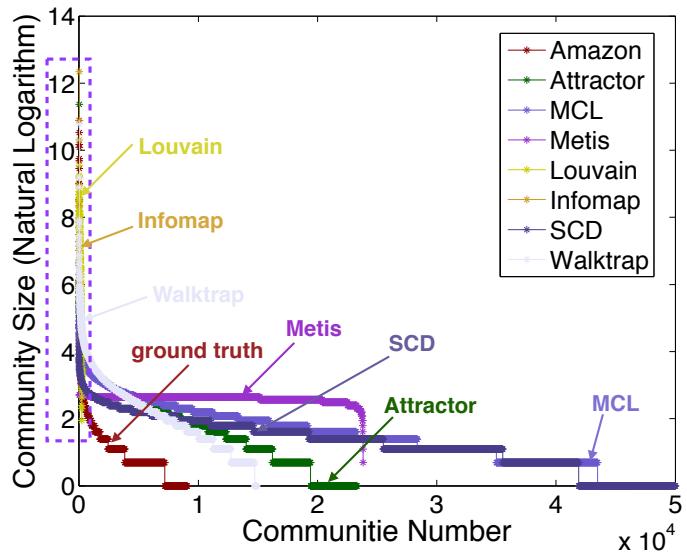


图 3-9 Amazon 网络上各算法得到的社团大小分布

3.4.3.2 噪声点检测

另外, 为了验证*Attractor* 是否可以有效地发现噪声点(奇异点/孤立点), 对于每个节点, 我们定义这个点的度与它的所有邻居的连边的条数之比, 为这个点的“局部噪声水平”, 一个节点的“局部噪声水平”放映了它的噪声程度。我们通过计算*Attractor* 算法找到的噪声点的“局部噪声水平”来衡量*Attractor* 处理噪声点的能力。作为比较, 我们计算了网络所有点的平均的“局部噪声水平”以及*MCL*^① 算法的结果, 见??。从结果中可以看到, *Attractor* 找到的噪声点的“局部噪声水平”明显低于同网络的平均的“局部噪声水平”, 也明显低于*MCL* 算法找到的噪声点的“局部噪声水平”。

3.4.4 时间复杂度测试

为了对比这些算法的运行时间, 我们生成了边数从10,000到100,000,000的具有平均度为20的LFR 网络, 结果见图3-11。我们可以发现*Attractor* 明显地快

① 选用*MCL* 算法作为比较的原因是, 其它算法受限于“Resolution limit”, 无法找到噪声点

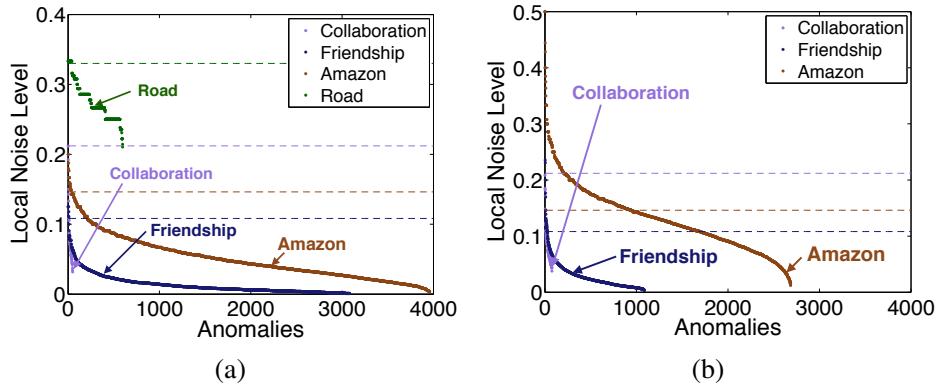


图 3-10 噪声检测实验结果

(a)Anomaly Attractor; (b)Anomaly MCL

于Modularity, Ncut 和MCL，并且Attractor 的运行时间与边数成正比($O(|E|)$)。同时，Attractor 要慢于Metis, Louvain 和Infomap，尽管许多数据集显示Attractor 的效果比这三种算法要好。

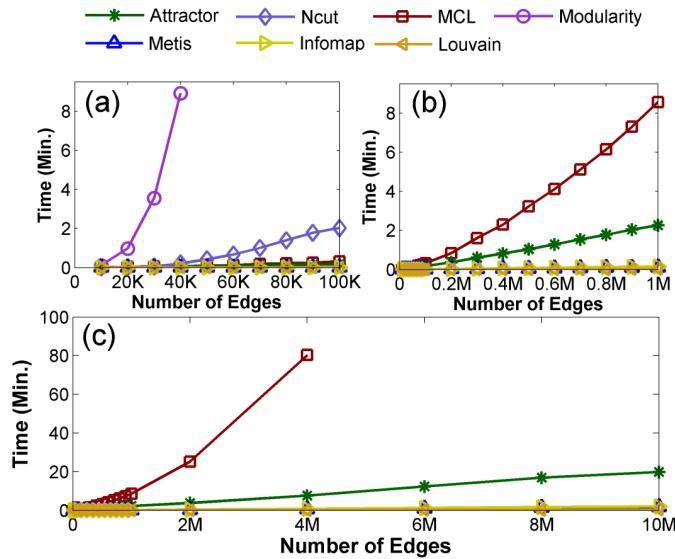


图 3-11 时间复杂度比较

3.5 总结

在这篇文章中，我们从动态交互角度出发，提出了一个新的社团挖掘算法Attractor，并且将Attractor 和经典的社团挖掘算法进行了比较，通过一系列的实验证明了Attractor 的优势。结果显示Attractor 并不受限于“resolution limit”，而是

可以高质量地检测出任意大小的社团，并且Attractor 时间复杂度与网络中边的条数成正比。这份工作只是一个开始，我们希望之后进一步改进Attractor 并以其为基础，研究网络科学中的其它问题。

第4章 出租车GPS轨迹聚类分析

之前, 第2章和第3章分别介绍了轨迹数据相似性度量以及图聚类算法(*Attractor* 算法). 接下来, 我们就利用这两个工作, 对一些出租车的GPS 轨迹进行聚类分析, 其流程如下: 利用第2 章中的方法确定出租车之间的相似度, 通过给定一个阈值将这些GPS 数据转化为相似网络^①; 之后利用3 中的*Attractor* 算法对这个相似网络进行聚类。在详细介绍聚类方法以及分析结果之前, 首先来说明一下数据集。

4.1 数据集说明

关于轨迹数据^②, 公开的数据集非常少。这里, 我们找到了Yu Zheng 等人在2011年发布的T-Drive 轨迹数据集^{[23][46]}的样例^③。这个数据集包含10,357辆出租车一周内的GPS采样数据。但是需要注意的是, 这些出租车采样的点的数量并不一致, 甚至数量相差很大(例如编号为1131的出租车有94939个采样点, 而编号为9424的出租车只有4393个采样点)。因此在这里, 我们选用了T-Drive 数据集第一个文件中的24辆出租车作为待测的轨迹数据, 它们之间采样点数量之间的差异不至于非常大。

4.2 聚类方法说明

假设这24 辆出租车GPS 数据编号为: 1, 2, ..., 24. 则对这些轨迹数据聚类过程可分为以下几步:

1. **计算这些轨迹之间的相似度** 利用2.3.2.1节中的方法将轨迹映射为行为网络, 之后根据2.3.2.2节中的公式(2-6) (2-7) (2-8) 计算这些出租车轨迹两两之间的距离。为之后聚类做准备。
2. **构建相似网络** 将每个轨迹对应于一个节点, 给定一个距离的阈值 ϵ ^④, 若轨迹 m 和轨迹 n 的距离小于这个阈值 ϵ 时, 则将 m 和 n 对应的节点相连, 构建相似网络。

① 网络的节点是出租车, 边代表出租车的距离

② 或是一般的含时行为数据

③ <http://research.microsoft.com/apps/pubs/?id=152883>

④ 在文章中, 取阈值 ϵ 的值为8

3.对轨迹进行聚类 利用3.3节中介绍的*Attractor* 算法对得到的相似网络进行聚类。

4.3 结果及讨论

这24辆出租车的距离结果如下：这24辆出租车一共被分成了2类，以及5个孤立点(见图4-1)。图4-1中不同颜色表示不同社团，其中，紫色社团中的节点彼此之间的联系非常紧密(边很多)，说明这些节点代表的轨迹之间的距离很小，彼此非常相似。而不同颜色的节点之间边很少，说明这些节点对应的轨迹彼此的距离较远，相似度较小。

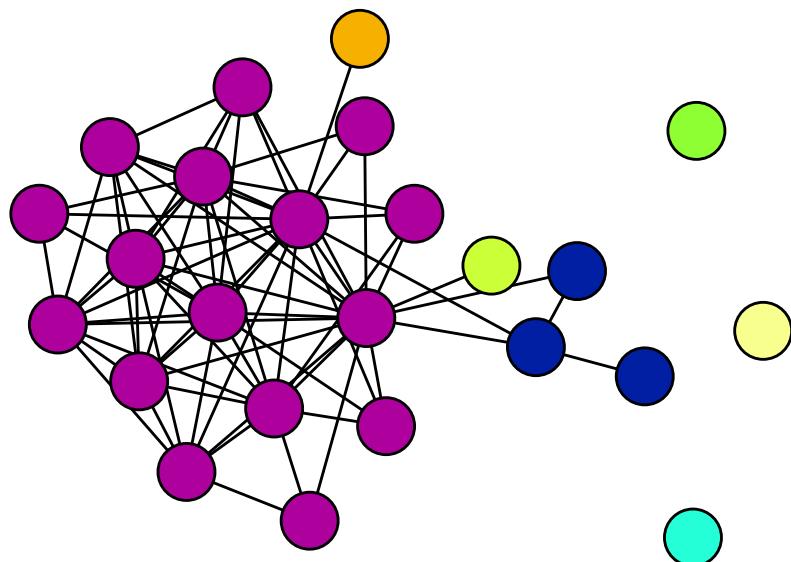


图 4-1 出租车轨迹聚类结果

以上结果虽然只是对24辆出租车聚类的结果，但这个案例展示了出租车轨迹数据聚类的流程。接下来，我们希望得到更多的数据集，在大规模的轨迹数据集上进行相似度以及聚类分析。

第5章 结语

这一章作为本文的最后一章，首先总结了这篇文章的主要工作，然后以上述工作为基础，讨论了接下来即将开始的工作。

5.1 本文工作总结

这篇文章以聚类分析为中心，以轨迹数据(行为数据)为应用对象，尝试提出了新的聚类算法以及相似度衡量方法，并将这两者用于轨迹数据，尝试挖掘轨迹数据中存在的规律。具体来说，本文第一部分(第2章)介绍了轨迹数据的相似度度量方法，第二部分(第3章)介绍了一个新的聚类算法，之后(第4章)将这两者应用于轨迹数据，尝试发现轨迹中存在的规律。

- 第一部分(第2章)介绍了一个新的衡量轨迹相似度的方法。它通过比较轨迹的时间特征和空间特征来确定轨迹之间的相似度。具体来说，它首先将每条轨迹转化为一个行为网络，通过赋予行为网络一些属性使之刻画轨迹的时间特征和空间特征，在此基础上，通过衡量各个轨迹对应的行为网络之间的相似度来确定轨迹之间的相似度。
- 第二部分(第3章)介绍了一个新的图聚类方法：*Attractor*。它的核心思想是将网络看作一个动态系统，通过交互过程边的距离的变化来发现社团。通过三种交互模式，*Attractor*可以有效地发现网络存在的社团结构。与经典的算法在大量数据集上的实验也说明了*Attractor*算法的优势。
- 第三部分(第4章)是出租车轨迹数据聚类的结果。首先使用2章中的相似度度量方法计算出了轨迹之间的相似度，之后通过给定的阈值，将这些轨迹转化为一个网络，网络中的每个节点对应一条轨迹，之后应用3章中介绍的图聚类算法*Attractor*对这些轨迹对应的网络进行了聚类。

5.2 下一步工作

这篇文章中提出的关于轨迹相似度的算法以及图聚类算法都是最初步的工作，以这两个工作为基础，还有许多有意思的内容可以深入挖掘。下面，我们就来介绍下接下来即将进行的工作。

对于轨迹相似度度量方法: 目前我们用行为网络的节点的度分布来表征轨迹的空间特征，用节点的时间分布来反映轨迹的时间特征，为了更好的把握轨迹的特征，接下来我们希望考虑更多的信息，例如轨迹的局部转移特征(对应于行为网络中边的信息)。

社团挖掘算法(*Attractor*): 这个社团挖掘算法仅仅是一个开端，在现有算法的基础上，还有许多问题可以挖掘，例如从理论上证明算法的收敛性(算法的理论基础)，或是通过并行运算缩短算法的运行时间，或是当网络中存在非常大的度的节点时怎样加速处理等等。接下来，我们希望能在这个算法的理论基础方面得到一些进展。

参考文献

- [1] C. C. Aggarwal, C. K. Reddy. Data clustering: algorithms and applications[M]. CRC Press, 2013
- [2] J. Han, M. Kamber, J. Pei. Data mining, southeast asia edition: Concepts and techniques[M]. Morgan kaufmann, 2006
- [3] I. H. Witten, E. Frank. Data mining: Practical machine learning tools and techniques[M]. Morgan Kaufmann, 2005
- [4] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, et al. Knowledge discovery and data mining: Towards a unifying framework.[M]. 1996, 82–88
- [5] J. Shao. Synchronization inspired data mining[M]. 2011
- [6] R. Agrawal, R. Srikant, et al. Fast algorithms for mining association rules[M]. 1994, 487–499
- [7] J. Han, J. Pei, Y. Yin. Mining frequent patterns without candidate generation[M]. 2000, 1–12
- [8] A. K. Jain, M. N. Murty, P. J. Flynn. Data clustering: a review[J]. ACM computing surveys (CSUR), 1999, 31(3):264–323
- [9] Z. Li, B. Ding, J. Han, et al. Swarm: Mining relaxed temporal moving object clusters[J]. Proceedings of the VLDB Endowment, 2010, 3(1-2):723–734
- [10] F. Giannotti, M. Nanni, F. Pinelli, et al. Trajectory pattern mining[M]. 2007, 330–339
- [11] Y. Zheng, X. Zhou. Computing with spatial trajectories[M]. Springer Science & Business Media, 2011
- [12] H. Wang, H. Su, K. Zheng, et al. An effectiveness study on trajectory similarity measures[M]. 2013, 13–22
- [13] C. Song, Z. Qu, N. Blumm, et al. Limits of predictability in human mobility[J]. Science, 2010, 327(5968):1018–1021
- [14] X.-Y. Yan, X.-P. Han, B.-H. Wang, et al. Diversity of individual mobility patterns and emergence of aggregated scaling laws[J]. Scientific reports, 2013, 3
- [15] E. J. Keogh, M. J. Pazzani. Derivative dynamic time warping.[M]. 2001, 5–7
- [16] P. Senin. Dynamic time warping algorithm review[J]. Information and Computer Science Department University of Hawaii at Manoa Honolulu, USA, 2008:1–23

- [17] S. Salvador, P. Chan. Toward accurate dynamic time warping in linear time and space[J]. Intelligent Data Analysis, 2007, 11(5):561–580
- [18] F. Itakura. Minimum prediction residual principle applied to speech recognition[J]. Acoustics, Speech and Signal Processing, IEEE Transactions on, 1975, 23(1):67–72
- [19] P. Chen, J. Gu, D. Zhu, et al. A dynamic time warping based algorithm for trajectory matching in lbs.[J]. International Journal of Database Theory & Application, 2013, 6(3)
- [20] D. Buzan, S. Sclaroff, G. Kollios. Extraction and clustering of motion trajectories in video[M]. 2004, 521–524
- [21] D. Jurafsky, J. H. Martin. Speech & language processing[M]. Pearson Education India, 2000
- [22] S. B. Needleman, C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins[J]. Journal of molecular biology, 1970, 48(3):443–453
- [23] J. Yuan, Y. Zheng, X. Xie, et al. Driving with knowledge from the physical world[M]. 2011, 316–324
- [24] S. Fortunato. Community detection in graphs[J]. Physics Reports, 2010, 486(3):75–174
- [25] J. Shi, J. Malik. Normalized cuts and image segmentation[J]. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2000, 22(8):888–905
- [26] S. Fortunato, M. Barthélemy. Resolution limit in community detection[J]. Proceedings of the National Academy of Sciences, 2007, 104(1):36–41
- [27] P. James, Y. Nadarajah, K. Haive, et al. Sustainable communities, sustainable development: Other paths for papua new guinea[J]. Hawaiian Journal of History, 2014, 48
- [28] S. E. Schaeffer. Graph clustering[J]. Computer Science Review, 2007, 1(1):27–64
- [29] Z. Wu, R. Leahy. An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation[J]. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 1993, 15(11):1101–1113
- [30] U. Von Luxburg. A tutorial on spectral clustering[J]. Statistics and computing, 2007, 17(4):395–416
- [31] M. E. Newman. Modularity and community structure in networks[J]. Proceedings of the National Academy of Sciences, 2006, 103(23):8577–8582
- [32] M. E. Newman. Fast algorithm for detecting community structure in networks[J]. Physical review E, 2004, 69(6):066133
- [33] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, et al. Fast unfolding of communities in large networks[J]. Journal of Statistical Mechanics: Theory and Experiment, 2008, 2008(10):P10008

- [34] M. Rosvall, C. T. Bergstrom. Maps of random walks on complex networks reveal community structure[J]. *Proceedings of the National Academy of Sciences*, 2008, 105(4):1118–1123
- [35] G. Karypis, V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs[J]. *SIAM Journal on scientific Computing*, 1998, 20(1):359–392
- [36] G. Karypis, V. Kumar. Multilevelk-way partitioning scheme for irregular graphs[J]. *Journal of Parallel and Distributed computing*, 1998, 48(1):96–129
- [37] S. Van Dongen. A cluster algorithm for graphs[J]. *Report-Information systems*, 2000(10):1–40
- [38] C. Hennig, B. Hausdorf. Design of dissimilarity measures: A new dissimilarity between species distribution areas[M]. Springer, 2006, 29–37
- [39] A. Strehl, J. Ghosh. Cluster ensembles—a knowledge reuse framework for combining multiple partitions[J]. *The Journal of Machine Learning Research*, 2003, 3:583–617
- [40] W. M. Rand. Objective criteria for the evaluation of clustering methods[J]. *Journal of the American Statistical association*, 1971, 66(336):846–850
- [41] A. Lancichinetti, S. Fortunato, F. Radicchi. Benchmark graphs for testing community detection algorithms[J]. *Physical review E*, 2008, 78(4):046110
- [42] W. W. Zachary. An information flow model for conflict and fission in small groups[J]. *Journal of anthropological research*, 1977:452–473
- [43] M. Girvan, M. E. Newman. Community structure in social and biological networks[J]. *Proceedings of the National Academy of Sciences*, 2002, 99(12):7821–7826
- [44] J. Yang, J. Leskovec. Defining and evaluating network communities based on ground-truth[J]. *Knowledge and Information Systems*, 2015, 42(1):181–213
- [45] J. Leskovec, J. Kleinberg, C. Faloutsos. Graph evolution: Densification and shrinking diameters[J]. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 2007, 1(1):2
- [46] J. Yuan, Y. Zheng, C. Zhang, et al. T-drive: driving directions based on taxi trajectories[M]. 2010, 99–108
- [47] Y. Yuan, M. Raubal. Similarity measurement of mobile phone user trajectories—a modified edit distance method[M]. 2012, 15–16
- [48] L. Chen, R. Ng. On the marriage of l_p -norms and edit distance[M]. 2004, 792–803
- [49] L. Chen, M. T. Özsü, V. Oria. Robust and fast similarity search for moving object trajectories[M]. 2005, 491–502

致 谢

在电子科技大学度过的大学时光是一段最值得我珍重的经历。借此机会，我要由衷地感谢带给我4年美好时光的电子科技大学以及在这里遇到的每一个关心和帮助过我的人。

首先，我要向我的导师邵俊明老师表达最真诚的谢意，邵老师对学术的追求以及认真负责的态度永远是我的学习榜样。我要诚挚地感谢容智海老师和周涛老师在各个方面给予我的无数的指导、帮助和关心。此外，我还要感谢互联网科学中心中每一位帮助过我的老师和同学，和你们相处的时间将会是美好的回忆。

我希望将诚挚的感谢送给那些认真给我们上课的老师以及帮助过我的其他老师们，你们让我在大学四年中学到了宝贵的知识。特别地，我希望向滕保华老师送上由衷的谢意，您对我的指导和帮助我将铭记于心。

我要感谢在电子科大认识的每一位同学、学长和学姐，大学四年学习和生活因为你们而丰富多采。我将永远记得四年中的经历过的数学建模、羽毛球比赛、教研室里写过程序、聚餐等等，这些将是我永久的记忆。特别地，我要感谢我的三位室友，非常幸运地和你们度过这4年。

最后，我想送给我的父母以及亲人们最特别的感激。你们无微不至的关心和支持陪伴了我20多年，你们无私的爱永远都是我人生中最好的礼物，你们带给了我一切。

附录 A Attractor 算法

A.1 Attractor 算法伪代码标示

Algorithm 1 Attractor

```

1: Input:  $G = (V, E, W)$ ,  $\lambda$ 
2: // Initialization of distances
3: for each edge  $e = \{u, v\} \in E$  do
4:   compute the initial distance  $d_e^0$  using Eq. (3.3.1);
5:   for each node  $x \in EN(u)$  do
6:     compute the distance  $d_{ux}^0$  using Eq. (3.3.1);
7:   end for
8:   for each node  $y \in EN(v)$  do
9:     compute the distance  $d_{vy}^0$  using Eq. (3.3.1);
10:  end for
11: end for

12: // Dynamic Interaction
13: Flag = TRUE;
14: while Flag do
15:   Flag = FALSE;
16:   for each edge  $e = \{u, v\} \in E$  do
17:     if  $0 < d_e^t < 1$  then
18:       Compute  $DI_e^t, CI_e^t, EI_e^t$  using Eq. (3-1), (3-2), (3-3);
19:        $\Delta d_e^t = DI_e^t + CI_e^t + EI_e^t$ ;
20:       if  $\Delta d_e^t \neq 0$  then
21:         // compute the renewable distance over time
22:          $d_e^{t+1} = d_e^t + \Delta d_e^t$ ;
23:         if  $d_e^{t+1} > 1$  then
24:            $d_e^{t+1} = 1$ ;
25:         end if
26:         if  $d_e^{t+1} < 0$  then
27:            $d_e^{t+1} = 0$ ;
28:         end if
29:         Flag = TRUE;
30:       end if
31:     end if
32:   end for
33: end while

34: //Find communities
35: for each edge  $e = \{u, v\} \in E$  do
36:   if  $d_e^{t+1} = 1$  then
37:     remove the edge  $e$  from the network;
38:   end if
39: end for
40: find the resulting components (communities)  $C$ ;
41: Output:  $C$ ;

```

附录 B 轨迹相似度度量

B.1 出租车轨迹相似度完整结果

表 B-1 Matrix of distances between node weight distribution

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	0.0	1.6	1.4	2.5	1.9	1.8	1.4	1.8	1.4	2.2	1.6	3.3	2.8	2.5	0.9	2.7	1.5	1.7	2.1	2.2	1.5	1.2	0.6	4.1
2	1.6	0.0	0.5	1.2	1.3	0.6	0.6	0.8	0.9	1.2	0.6	1.7	2.3	1.4	0.5	1.2	1.2	1.8	0.9	1.2	1.1	14	1.0	4.3
3	1.4	0.5	0.0	1.0	1.1	0.4	0.4	0.5	0.7	0.8	0.3	1.3	1.7	1.1	0.2	1.2	0.9	1.6	0.7	0.8	0.9	11	0.7	4.2
4	2.5	1.2	1.0	0.0	1.7	1.3	1.0	1.0	1.2	1.8	1.0	1.9	2.3	1.9	1.1	2.0	1.7	2.7	1.5	1.3	1.4	2.1	0.2	4.8
5	1.9	1.3	1.1	1.7	0.0	1.3	1.0	1.1	0.8	2.1	1.4	2.8	1.7	1.6	1.0	2.3	1.5	2.3	1.7	1.7	1.0	18	1.1	4.9
6	1.8	0.6	0.4	1.3	1.3	0.0	0.6	0.9	1.1	1.1	0.6	1.7	2.2	1.7	0.5	1.2	1.2	2.0	0.9	1.2	1.2	16	1.1	4.2
7	1.4	0.6	0.4	1.0	1.0	0.6	0.0	0.6	0.8	1.0	0.5	1.5	2.0	1.4	0.4	1.3	1.0	1.9	0.9	1.0	0.9	13	0.9	3.2
8	1.8	0.8	0.5	1.0	1.1	0.9	0.6	0.0	0.8	1.2	0.5	1.6	1.8	1.0	0.6	1.6	1.1	1.8	1.2	1.1	0.9	1.4	0.8	4.2
9	1.4	0.9	0.7	1.2	0.8	1.1	0.8	0.8	0.0	1.5	0.8	2.0	1.4	1.2	0.6	2.0	1.1	2.1	1.5	1.0	0.7	12	0.6	3.7
10	2.2	1.2	0.8	1.8	2.1	1.1	1.0	1.2	1.5	0.0	0.9	1.8	3.0	2.3	0.7	1.7	1.6	2.5	1.5	1.5	1.9	18	1.3	4.2
11	1.6	0.6	0.3	1.0	1.4	0.6	0.5	0.5	0.8	0.9	0.0	1.4	2.1	1.3	0.4	1.3	1.1	1.8	1.0	1.0	1.1	13	0.8	4.2
12	3.3	1.7	1.3	1.9	2.8	1.7	1.5	1.6	2.0	1.8	1.4	0.0	3.1	2.6	1.5	2.2	2.3	3.3	1.7	1.7	2.4	28	1.8	5.5
13	2.8	2.3	1.7	2.3	1.7	2.2	2.0	1.8	1.4	3.0	2.1	3.1	0.0	2.7	1.9	3.1	2.4	3.3	2.4	1.8	1.9	25	1.9	6.0
14	2.5	1.4	1.1	1.9	1.6	1.7	1.4	1.0	1.2	2.3	1.3	2.6	2.7	0.0	1.0	2.2	1.5	2.7	1.9	1.8	1.2	20	1.4	5.2
15	0.9	0.5	0.2	1.1	1.0	0.5	0.4	0.6	0.6	0.7	0.4	1.5	1.9	1.0	0.0	1.2	0.8	1.4	0.9	1.0	0.8	8	0.7	3.8
16	2.7	1.2	1.2	2.0	2.3	1.2	1.3	1.6	2.0	1.7	1.3	2.2	3.1	2.2	1.2	0.0	2.0	2.6	1.5	1.9	2.2	24	1.9	4.5
17	1.5	1.2	0.9	1.7	1.5	1.2	1.0	1.1	1.1	1.6	1.1	2.3	2.4	1.5	0.8	2.0	0.0	1.8	1.5	1.6	1.2	1.2	0.2	4.6
18	1.7	1.8	1.6	2.7	2.3	2.0	1.9	1.8	2.1	2.5	1.8	3.3	3.3	2.7	1.4	2.6	1.8	0.0	2.1	2.5	1.8	9	1.7	4.5
19	2.1	0.9	0.7	1.5	1.7	0.9	0.9	1.2	1.5	1.5	1.0	1.7	2.4	1.9	0.9	1.5	1.5	2.1	0.0	1.4	1.6	1.8	0.4	4.2
20	2.2	1.2	0.8	1.3	1.7	1.2	1.0	1.1	1.0	1.5	1.0	1.7	1.8	1.8	1.0	1.9	1.6	2.5	1.4	0.0	1.3	2.0	0.2	4.7
21	1.5	1.1	0.9	1.4	1.0	1.2	0.9	0.9	0.7	1.9	1.1	2.4	1.9	1.2	0.8	2.2	1.2	1.8	1.6	1.3	0.0	13	0.9	4.5
22	1.2	1.4	1.1	2.1	1.8	1.6	1.3	1.4	1.2	1.8	1.3	2.8	2.5	2.0	0.8	2.4	1.2	0.9	1.8	2.0	1.3	0	1.2	4.6
23	1.6	1.0	0.7	1.2	1.1	1.1	0.9	0.8	0.6	1.3	0.8	1.8	1.9	1.4	0.7	1.9	1.2	1.7	1.4	1.2	0.9	1.2	0.0	4.4
24	4.1	4.3	4.2	4.8	4.9	4.2	3.2	4.2	3.7	4.2	4.2	5.5	6.0	5.2	3.8	4.5	4.6	4.5	4.2	4.7	4.5	4.6	0.4	0.0

表 B-2 Matrix of distances between visited time distribution

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	0.0	7.3	7.8	7.9	8.4	7.6	8.4	7.7	7.9	8.4	7.7	8.3	7.8	8.4	6.4	8.9	7.6	8.5	8.6	8.4	6.8	6.9	8.1	9.4
2	7.3	0.0	5.7	7.8	7.4	5.6	6.7	7.6	6.8	7.5	6.3	7.3	7.9	7.3	5.7	7.3	7.3	7.1	7.9	7.1	6.3	7.3	6.3	9.3
3	7.8	5.7	0.0	6.8	8.0	6.1	7.1	7.4	7.0	6.6	6.0	7.0	8.1	7.5	7.4	6.4	6.8	7.6	7.4	7.5	7.5	6.5	6.3	9.0
4	7.9	7.8	6.8	0.0	8.0	8.2	7.7	7.7	6.8	7.3	6.9	7.5	8.7	8.0	7.5	8.0	7.7	9.1	7.2	7.7	8.1	6.8	7.6	9.7
5	8.4	7.4	8.0	8.0	0.0	7.3	8.0	7.9	6.0	8.4	8.1	8.9	7.6	8.0	7.2	7.9	7.7	9.0	8.0	7.3	7.0	8.3	8.2	9.2
6	7.6	5.6	6.1	8.2	7.3	0.0	6.6	7.2	7.4	7.9	7.0	7.8	8.8	7.3	6.4	7.5	7.6	6.3	7.0	8.0	7.1	7.6	6.8	9.8
7	8.4	6.7	7.1	7.7	8.0	6.6	0.0	7.8	7.6	8.3	7.5	7.3	8.9	8.0	6.3	8.4	7.8	7.6	7.7	7.8	8.3	7.2	8.0	9.0
8	7.7	7.6	7.4	7.7	7.9	7.2	7.8	0.0	7.5	8.4	7.5	8.9	8.5	8.3	7.7	7.9	8.5	8.4	7.5	8.2	7.5	7.5	8.0	9.1
9	7.9	6.8	7.0	6.8	6.0	7.4	7.6	7.5	0.0	8.5	7.9	8.0	7.8	7.7	6.5	8.4	7.4	8.6	8.4	7.9	6.8	6.9	7.4	9.4
10	8.4	7.5	6.6	7.3	8.4	7.9	8.3	8.4	8.5	0.0	7.4	8.5	8.3	8.7	8.4	8.0	8.7	8.5	7.9	8.0	8.9	7.8	8.0	8.6
11	7.7	6.3	6.0	6.9	8.1	7.0	7.5	7.5	7.9	7.4	0.0	7.6	7.9	6.8	7.0	7.5	7.4	7.9	7.0	7.2	7.4	7.3	6.9	8.8
12	8.3	7.3	7.0	7.5	8.9	7.8	7.3	8.9	8.0	8.5	7.6	0.0	9.2	7.9	7.8	8.5	8.9	9.2	7.2	7.5	8.2	8.5	8.2	9.7
13	7.8	7.9	8.1	8.7	7.6	8.8	8.9	8.5	7.8	8.3	7.9	9.2	0.0	8.3	8.7	8.9	8.8	9.4	9.2	8.3	8.9	8.6	8.1	9.6
14	8.4	7.3	7.5	8.0	8.0	7.3	8.0	8.3	7.7	8.7	6.8	7.9	8.3	0.0	7.0	8.7	7.7	8.2	7.3	8.0	7.6	7.4	7.2	9.7
15	6.4	5.7	7.4	7.5	7.2	6.4	6.3	7.7	6.5	8.4	7.0	7.8	8.7	7.0	0.0	7.2	6.0	6.9	6.6	7.3	5.9	6.7	6.5	8.8
16	8.9	7.3	6.4	8.0	7.9	7.5	8.4	7.9	8.4	8.0	7.5	8.5	8.9	8.7	7.2	0.0	8.0	7.5	8.3	7.9	7.4	8.3	8.1	8.4
17	7.6	7.3	6.8	7.7	7.7	7.6	7.8	8.5	7.4	8.7	7.4	8.9	8.8	7.7	6.0	8.0	0.0	8.9	8.3	7.8	7.1	6.9	7.7	9.3
18	8.5	7.1	7.6	9.1	9.0	6.3	7.6	8.4	8.6	8.5	7.9	9.2	9.4	8.2	6.9	7.5	8.9	0.0	8.8	9.7	9.1	8.0	8.2	9.7
19	8.6	7.9	7.4	7.2	8.0	7.0	7.7	7.5	8.4	7.9	7.0	7.2	9.2	7.3	6.6	8.3	8.8	0.0	7.3	7.3	7.7	8.0	8.1	
20	8.4	7.1	7.5	7.7	7.3	8.0	7.8	8.2	7.9	8.0	7.2	7.5	8.3	8.0	7.3	7.9	7.8	9.7	7.3	0.0	8.1	7.6	7.1	9.5
21	6.8	6.3	7.5	8.1	7.0	7.1	8.3	7.5	6.8	8.9	7.4	8.2	8.9	7.6	5.9	7.4	7.1	9.1	7.3	8.1	0.0	7.3	6.9	8.7
22	6.9	7.3	6.5	6.8	8.3	7.6	7.2	7.5	6.9	7.8	7.3	8.5	8.6	7.4	6.7	8.3	6.9	8.0	7.7	7.6	7.3	0.0	7.2	8.8
23	8.1	6.3	6.3	7.6	8.2	6.8	8.0	8.0	7.4	8.0	6.9	8.2	8.1	7.2	6.5	8.1	7.7	8.2	8.0	7.1	6.9	7.2	0.0	9.4
24	9.4	9.3	9.0	9.7	9.2	9.8	9.0	9.1	9.4	8.6	8.8	9.7	9.6	9.7	8.8	8.4	9.3	9.7	8.1	9.5	8.7	8.8	9.4	0.0

表 B-3 Matrix of distances between taxi trajectories

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	
1	0.0	11.6	11.1	19.4	16.4	13.9	11.7	14.0	10.9	18.6	12.6	27.6	21.9	21.3	6.1	24.1	11.5	14.5	18.5	18.5	10.4	7.9	12.6	38.8	
2	11.6	0.0	2.9	9.7	9.6	3.4	3.8	6.1	6.4	8.7	3.9	12.3	18.0	10.2	2.6	9.1	8.4	12.8	7.3	8.1	6.8	10.3	6.1	40.1	
3	11.1	2.9	0.0	6.6	8.9	2.7	2.8	3.6	5.2	5.4	2.0	8.8	14.0	8.5	1.8	7.5	6.4	11.8	5.4	6.1	6.5	7.1	4.4	37.6	
4	19.4	9.7	6.6	0.0	13.9	10.5	7.7	7.8	8.4	13.1	7.0	13.9	19.9	14.9	8.1	15.6	13.1	24.6	10.7	10.3	11.4	14.3	9.3	46.9	
5	16.4	9.6	8.9	13.9	0.0	9.4	8.3	8.5	4.7	17.5	11.1	24.7	12.8	12.6	7.3	18.2	11.4	20.5	14.0	12.4	6.7	14.6	9.3	45.5	
6	13.9	3.4	2.7	10.5	9.4	0.0	3.9	6.5	8.4	8.9	4.5	13.4	19.7	12.2	3.3	9.3	9.4	12.6	6.1	9.5	8.4	12.4	7.7	41.2	
7	11.7	3.8	2.8	7.7	8.3	3.9	0.0	4.6	6.0	8.6	3.4	11.1	18.1	11.1	2.3	11.1	7.4	14.2	6.8	8.1	7.8	9.5	7.0	29.3	
8	14.0	6.1	3.6	7.8	8.5	6.5	4.6	0.0	5.9	10.0	4.0	14.3	15.4	8.4	4.8	12.6	9.3	15.1	8.9	9.2	6.6	10.4	6.5	38.6	
9	10.9	6.4	5.2	8.4	4.7	8.4	6.0	5.9	0.0	12.5	6.6	16.1	11.3	9.5	3.6	16.8	7.9	18.0	12.6	7.5	4.7	8.4	4.6	34.9	
10	18.6	8.7	5.4	13.1	17.5	8.9	8.6	10.0	12.5	0.0	6.7	15.6	25.2	20.2	5.6	13.2	13.8	21.2	12.2	12.0	16.7	13.9	10.8	36.5	
11	12.6	3.9	2.0	7.0	11.1	4.5	3.4	4.0	4.0	6.6	6.7	0.0	11.0	16.6	8.6	2.7	9.6	8.1	14.0	7.0	7.0	8.5	9.5	5.8	37.1
12	27.6	12.3	8.8	13.9	24.7	13.4	11.1	14.3	16.1	15.6	11.0	0.0	28.2	20.4	11.8	18.6	20.2	30.6	12.3	12.8	20.0	23.6	14.9	52.9	
13	21.9	18.0	14.0	19.9	12.8	19.7	18.1	15.4	11.3	25.2	16.6	28.2	0.0	22.8	16.9	27.9	21.2	30.8	21.9	15.4	16.6	21.5	15.0	57.4	
14	21.3	10.2	8.5	14.9	12.6	12.2	11.1	8.4	9.5	20.2	8.6	20.4	22.8	0.0	7.3	19.2	11.4	21.8	14.8	14.1	14.3	9.3	14.8	9.9	50.6
15	6.1	2.6	1.8	8.1	7.3	3.3	2.3	4.8	3.6	5.6	2.7	11.8	16.9	7.3	0.0	8.7	4.8	9.5	5.8	7.1	5.1	4.5	33.4		
16	24.1	9.1	7.5	15.6	18.2	9.3	11.1	12.6	16.8	13.2	9.6	18.6	27.9	19.2	8.7	0.0	15.7	19.4	12.8	15.3	15.9	19.9	15.1	38.1	
17	11.5	8.4	6.4	13.1	11.4	9.4	7.4	9.3	7.9	13.8	8.1	20.2	21.2	11.4	4.8	15.7	0.0	15.7	12.7	12.7	8.5	8.5	9.4	42.4	
18	14.5	12.8	11.8	24.6	20.5	12.6	14.2	15.1	18.0	21.2	14.0	30.6	30.8	21.8	9.5	19.4	15.7	0.0	18.6	24.4	16.0	7.2	13.7	43.1	
19	18.5	7.3	5.4	10.7	14.0	6.1	6.8	8.9	12.6	12.2	7.0	12.3	21.9	14.1	5.8	12.8	12.7	18.6	0.0	10.0	11.4	13.6	11.5	34.3	
20	18.5	8.1	6.1	10.3	12.4	9.5	8.1	9.2	7.5	12.0	7.0	12.8	15.4	14.3	7.1	15.3	12.7	24.4	10.0	0.0	10.9	14.9	8.8	44.0	
21	10.4	6.8	6.5	11.4	6.7	8.4	7.8	6.6	4.7	16.7	8.5	20.0	16.6	9.3	5.1	15.9	8.5	16.0	11.4	10.9	0.0	9.4	5.9	39.0	
22	7.9	10.3	7.1	14.3	14.6	12.4	9.5	10.4	8.4	13.9	9.5	23.6	21.5	14.8	5.1	19.9	8.5	7.2	13.6	14.9	9.4	0.0	8.9	40.8	
23	12.6	6.1	4.4	9.3	9.3	7.7	7.0	6.5	4.6	10.8	5.8	14.9	15.0	9.9	4.5	15.1	9.4	13.7	11.5	8.8	5.9	8.9	0.0	40.9	
24	38.8	40.1	37.6	46.9	45.5	41.2	29.3	38.6	34.9	36.5	37.1	52.9</													