# 浙 江 大 學

# 本 科 生 毕 业 论 文（设计）


题目 一种基于回溯方式的时空同位模式挖掘算法

姓名与学号 殷力昂 3053021137

指导教师 何钦铭 教授

年级与专业 计算机科学与技术 0504

所在学院 计算机科学与技术学院

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


# TITLE: Spatio-temporal Co-location Pattern Mining: A Look-back Approach

Author: Li'ang Yin

Supervisor: Prof. Qinming He

Subject: Computer Science and Technology

College: College of Computer Science

Submitted Date: 2009.6.5

# 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

时空同位模式是在空间和时间邻域中频繁出现在一起的特征集合。基于回溯方式的时空同位模式挖掘在空间同位模式定义的基础上，向前回溯若干时间片，从而能发现更多有意义的同位模式。时空同位模式挖掘可以有效地发现气候（特征变化）、军事（战场计划和策略）、生态（跟踪物种和污染物的迁移）和运输（道路网络设计）等多种领域相关的同位模式。本文在分析了时空同位模式挖掘相关工作的基础上，引入了一个新颖的时空同位模式流行度衡量标准，从而建立了基于回溯方式的时空同位模式模型，并提出了一种基于回溯方式的时空同位模式挖掘算法（回溯算法）。本文分析和证明了所提出的流行度衡量标准的单调递减特性，以及回溯算法的完备性和正确性，并基于合成数据集和真实数据集的实验分析，评估了回溯算法与其他时空同位模式挖掘算法在效果和性能上的区别，验证了回溯算法在时空同位模式发现数量和运行效率上的优势。

**关键词** 空间同位模式、时空同位模式、时空数据挖掘

# Abstract

Spatio-temporal co-location patterns represent subsets of objects-types which are located together in space and time. Look-back spatio-temporal co-location patterns (LBCLPs) mining are based on current spatial co-location patterns, and look-back some time slots to discover more co-location patterns. LBCLPs mining is useful to find patterns of climate, tactics, ecology and transport. We propose a novel prevalence measurement to model LBCLPs and a LBCLP mining algorithm in this paper. This paper shows that the novel prevalence measurement is monotonically non-increasing, and LBCLP mining algorithm is complete and correct. Compared to other spatio-temporal co-location pattern mining algorithms, we show that the LBCLP mining algorithm can discover more spatio-temporal co-location patterns and has reasonable execution time through experiments.

**Keywords** spatial co-location patterns, spatio-temporal co-location patterns, spatio-temporal data mining

# 目录

摘要 ..... I
Abstract ..... II
目录 ..... III
第 1 章 绪论 ..... 1
1.1 课题背景 ..... 1
1.2 相关工作 ..... 1
1.3 创新点 ..... 2
1.4 章节安排 ..... 3
第 2 章 同位模式基本概念 ..... 4
2.1 同位模式问题的描述 ..... 4
2.2 空间同位模式和空间同位规则的形式化表述 ..... 5
2.3 本章小结 ..... 7
第 3 章 基于回溯方式的时空同位模式挖掘概念 ..... 8
3.1 基于回溯方式的时空同位模式概念描述 ..... 8
3.2 基于回溯方式的时空同位模式挖掘概念的形式化表述 ..... 9
3.3 基于回溯方式的时空同位模式建模 ..... 10
3.4 本章小结 ..... 14
第 4 章 基于回溯方式的时空同位模式挖掘算法 ..... 15
4.1 算法描述 ..... 15
4.1.1 输入、输出和变量 ..... 15
4.1.2 算法步骤 ..... 17
4.2 算法细节 ..... 19
4.2.1 实例列的结构 ..... 19
4.2.2 流行度剪枝 ..... 20
4.2.3 全局流行度剪枝 ..... 21
4.3 本章小结 ..... 21

第 5 章 基于回溯方式的时空同位模式挖掘算法分析.....22
5.1 算法单调递减特性、完备性和正确性证明.....22
5.1.1 单调递减特性证明.....22
5.1.2 完备性和正确性证明.....23
5.1.3 算法单调递减特性和完备性的意义.....24
5.2 算法的计算复杂度分析.....25
5.2.1 全关系情形.....25
5.2.2 一般情形.....27
5.3 算法扩展.....27
5.3.1 扩展回溯算法为混合驱动模式挖掘算法.....27
5.3.2 扩展回溯算法用于从空间同位模式到时空同位模式的挖掘.....28
5.4 本章小结.....29
第 6 章 基于回溯方式的时空同位模式挖掘算法实验分析.....30
6.1 合成数据生成方法.....30
6.1.1 用于回溯方式和混合驱动模式的数据合成方式.....30
6.1.2 用于直接扩展算法的数据合成方式.....32
6.2 合成数据实验及分析.....33
6.2.1 不同算法挖掘效果的实验及分析.....33
6.2.2 不同算法执行效率的实验及分析.....35
6.2.3 回溯算法挖掘效果和执行效率的实验及分析.....40
6.3 真实数据实验.....44
6.3.1 真实数据和使用方法说明.....44
6.3.2 算法挖掘效果分析.....45
6.3.3 算法执行效率分析.....47
6.4 本章小结.....48
第 7 章 总结及下一阶段研究方向.....50
参考文献.....51
致谢.....52

# 第1章 绪论

## 1.1 课题背景

给出一个具有各种不同空间特征（Feature）的实例的集合，通过同位模式（Co-location Patterns）的挖掘算法，可以找出在空间上频繁出现在一起的同位特征的集合。例如，通过对生态数据集的分析可以发现共生的生物种群。扩展空间的概念，把时间也作为一个维度引入到数据集合当中，就有了挖掘时空同位模式（Spatio-temporal Co-location Patterns）的需要。通过在一个数据集中应用时空同位模式的挖掘算法，可以有效地发现可以发现气候（特征变化）、军事（战场计划和策略）、生态（跟踪物种和污染物的迁移）和运输（道路网络设计）等多方面的有用模式。

目前对于时空同位模式的挖掘算法通常只孤立地处理每一个时间片（Time Slot），忽略了时间片之间的关联性，因而可能丢失一些有用的模式。例如，一种部队追及的模式，若使用只对当前时间片进行分析的方法，如采用混合驱动模式算法[7]，在多数的时间片上，上述的模式都不能成为同位模式，因而被剪枝丢弃，之后对整个时间轴进行分析的时候，由于这种同位模式所占比例很小，所以不能被发现。

本文主要针对这个问题，考虑了时间片之间的相关性，提出了基于回溯方式的时空同位模式挖掘算法。回溯算法不是孤立地分析每一个时间片，而是把几个邻近的时间片叠加起来考虑，这样就保留了时间片之间关联性，因此能够挖掘出更多有用的时空同位模式。

## 1.2 相关工作

同位模式（Co-location）的概念首先由 Yan Huang 等提出[1]，同位模式（Co-location Patterns）代表了那些在空间上形成关联的特征。这些特征的实例（Instance）在空间中频繁地出现在一起。通过文献[1]的同位模式挖掘算法（该算法有单调递减特性），可以从空间数据集中发现有用的同位模式。该文献同时给出了算法正确性和完备性的证明。文献[3]和[5]对于同位模式挖掘算法的计算复杂度做出了详细的分析，指出在同位模式挖掘算法中，连接（Join）的部分花

费了大量的时间。这两篇论文分别采用了部分连接（Partial Join）和无连接（Join-less）的改进算法来减少同位模式挖掘算法在连接部分花费的时间，在论文给出的实验结果中，对于实例比较稠密的数据集，经过改进的算法整体效率确实有了很大提高。

文献[6]、[7]、[8]、[11]对空间同位模式概念进行扩展，使得扩展之后的同位模式概念能应用于时空数据集。这几篇文献基本的思想，是对每一个时间片独立地实行同位模式挖掘算法，计算出各自时间片上同位模式的参与度（Participation Index），然后沿着时间轴排列这些参与度，这样就得到以时间为横轴，同位模式的参与度为纵轴的时间——参与度曲线，之后再根据不同的应用需求对这条曲线进行分析：在文献[6]中是与用户输入的需求曲线比较，文献[10]中考察这条曲线是否是单调递增的，文献[7]和[8]考虑这条曲线在一个给定的阈值之上的部分占整条时间轴的比例，从而得到有用的时空同位模式。这些算法孤立地分析每个时间片，忽略了时间片之间的关联，因此可能丢失有用的时空同位模式（尽管算法被证明具有正确性和完备性）。

时空序列模式挖掘算法[9]考察具有连续空间和时间信息的特征的实例，对相继发生在同一区域某些特征的实例进行分析，进而发现特征之间的链式关系。同时，该挖掘算法采用了密度比作为衡量特征间相互关系的方法，给同位模式挖掘提供了更多的衡量办法。该方法考虑到了时空序列模式在空间上的连续演化特性，但由于所提出的模式衡量标准缺乏一定的单调递减特性，因此，在算法复杂度上有一定缺陷。

## 1.3 创新点

本文针对目前时空同位模式挖掘算法存在的问题（忽略不同时间片之间的关联性），提出了一种基于回溯方式的时空同位模式挖掘算法。该算法考虑了不同时间片之间的关联，引入了“回溯”这一反映时间片关联性的参数（当不采用回溯时，即成为文献[7]的算法），较目前的挖掘算法可以发现更多的时空同位模式。同时，该算法具有单调递减特性，能有效降低计算复杂度。本文的创新点主要有以下几个：

1. 总结了时空同位模式挖掘问题和相关的基本概念；
2. 提出了一个新颖的时空同位模式流行度衡量标准，并在此基础上建立了

一种基于回溯方式的时空同位模式挖掘模型；

3. 基于上述模型，提出了一种基于回溯方式的时空同位模式挖掘算法；

4. 证明了上述流行度衡量标准的单调递减特性，以及回溯算法的正确性和完备性；

5. 基于合成数据集与真实数据集对回溯算法和其他时空同位模式挖掘算法的效果和性能进行评估和分析。

## 1.4 章节安排

第二章详述了基本的同位模式概念和挖掘方法。第三章提出了基于回溯方式的时空同位模式挖掘概念，并给出了形式化的表述和定义。后一章给出了基于回溯方式的时空同位模式挖掘算法和交代了算法的细节。第五章分析该算法的时间复杂度和证明算法的一些性质，并且给出了扩展该算法用于其他同位模式挖掘的方法。实验的分析在第六章。最后是总结和下一阶段研究方向。

# 第2章 同位模式基本概念

## 2.1 同位模式问题的描述

我们在处理空间数据集时，经常能够发现几种不同特征的实例（Instance）集中在一起。如观察分析一个生态系统，就有可能发现几种生物种群有共生关系或者生活在同一个区域。特征间的相互关系表现在空间数据上，是这些特征的实例在空间坐标位置上比较靠近（通常是欧式距离），这就引出了同位模式的概念。

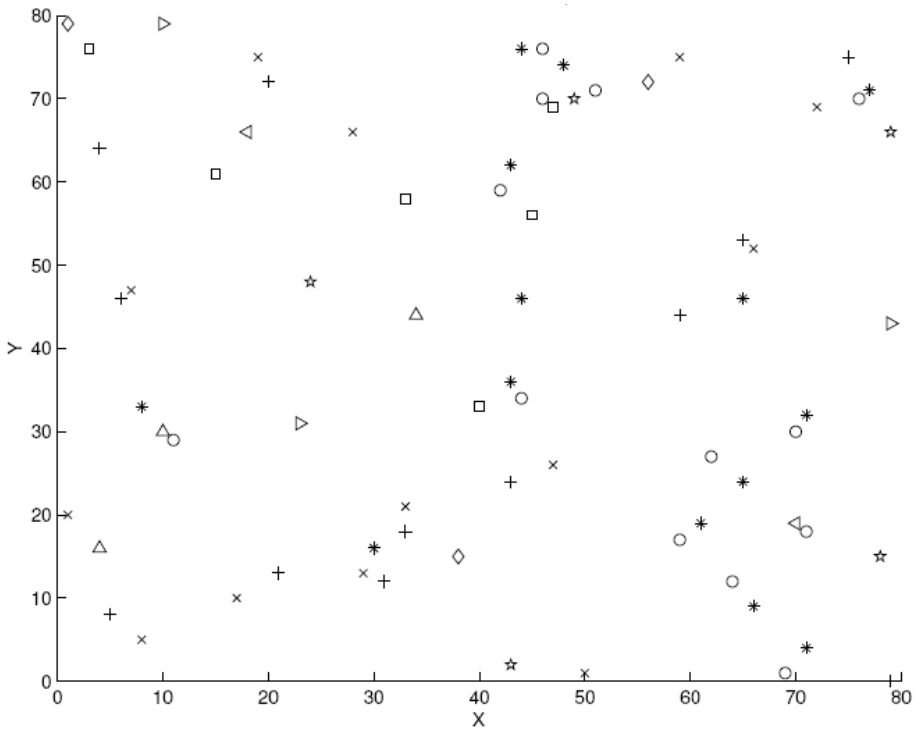



图 2.1 空间同位模式的例子（摘自文献[1]）

同位模式（Co-location Pattern）[1]代表了那些在空间上形成关联的特征。这些特征的实例在空间中频繁地出现在一起。图 2.1 展示了一个二维空间数据集，里面包含几种特征的实例，如“＋”、“×”、“∘”等，每一个图形代表了一个该图形对应特征的实例。仔细观察，我们会发现两组同位模式，分别是{'+','×'}和{'∘','*'}，他们在空间中经常伴随出现。在真实世界中，该同位模式可能是生态系统中生活着的尼罗河鳄鱼和埃及千鸟（他们是共生的一个典型例子）。同位模

式挖掘算法主要用于找出这些可能的同位模式。

## 2.2 空间同位模式和空间同位规则的形式化表述

对于空间同位模式挖掘问题，形式化的表述如下：

给出：

1) 包含$K$个特征的集合$F$，$F=\{f_1, f_2, ... f_k\}$。
2) 特征的实例的集合$I$，$I=\{i_1, i_2, ... i_n\}$，其中，$i_i$是一个向量，可以写成$<instance_id, spatial_feature_type, location>$，$instance_id$是该实例的唯一标识，$spatial_feature_type \in F$是该实例所属的空间特征，$location$是实例在空间中的坐标。
3) 一个近邻关系（Proximity Neighborhood）$R$用来描述实例之间的邻域关系，即两个实例靠得多近才能认为是有关系的。
4) 流行度的阈值$\theta$（Prevalence Threshold）和条件概率（Conditional Probability）的阈值$\alpha$。

同位模式挖掘算法的目的是找出所有流行（Prevalent）的同位模式。

同位模式（Co-location Pattern）是空间特征集合的子集，如特征组成的集合$\{f_1, f_2, f_3\}$就是一个同位模式。文献[1]采用的同位规则（Co-location Rule）形式如下：$c_1 \Rightarrow c_2(p, cp)$，这里$c_1$和$c_2$都是同位模式，并且$c_1 \bigcap c_2 = \varnothing$，$p$是一个流行度参数（Prevalence），$cp$是在$c_2$的条件下$c_1$发生的条件概率（Conditional Probability），满足$p \geq \theta$并且$cp \geq \alpha$。

近邻关系$R$用于考察分别属于不同特征的实例是否存在相互间的关系。近邻关系$R$具有自反对称性质，典型近邻关系$R$的例子是使用欧式距离作为衡量手段，当两个实例之间距离小于或等于一个给定的距离阈值$D$时，两者就在近邻关系$R$下形成了近邻关系，当处理三个或三个以上实例时，要求每一个实例对其他所有的实例之间的距离都不大于距离阈值$D$，即这些实例形成团（Clique）[4]。近邻关系$R$是作为同位模式挖掘算法的输入由用户给出的，在给出近邻关系$R$时应该基于应用领域的需要。

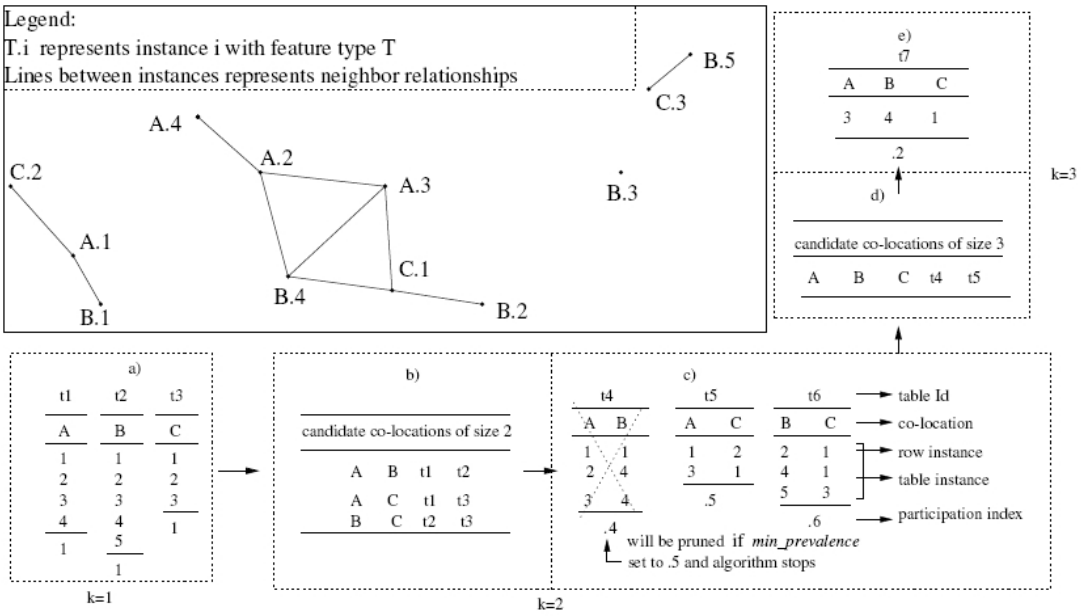



图 2.2 通过特征的实例生成实例表（摘自文献[1]）

如果有一个满足近邻关系 $R$ 的实例集合 $I$，如图 2.2 中的{A.3,B.4,C.1}，它包含了特征 $\{A,B,C\}$，而 $I$ 的所有真子集都不能全部包含这些特征，则称 $I$ 是一个实例列（Row Instance，写作 $row\_instance(c)$，其中 $c$ 是一个同位特征）。例如，图 2.2 中的{A.3,B.4,C.1}是同位模式 $\{A,B,C\}$ 的一个实例列。但是{A.2,A.3,B.4,C.1}不能作为{A,B,C}的一个实例列，因为{A.2,A.3,B.4,C.1}的一个真子集{A.2,B.4,C.1}包含了全部特征的实例。实例列的另一个例子是{A.2,A.4}不能作为{A}的实例列，因为它的真子集{A.2}或者{A.4}包含了{A}的全部特征 A。对实例列的一个简单理解，可以认为一个实例列中不能出现重复的特征。一个同位模式 $c$ 的实例表（Table Instance，写作 $table\_instance(c)$）是该同位特征的全部实例列的集合。在图 2.2 中，$t_1$，$t_2$，$t_3$，$t_4$，$t_5$，$t_6$ 和 $t_7$ 表示了实例表。举例来说，$t_5=\{\{A.1,C.2\},\{A.3,C.1\}\}$ 是同位特征{A,C}的实例表。

参与比（Participation Ratio）$pr(c,f_i)$ 是特征 $f_i$ 在同位特征 $c$ 中的实例数量与特征 $f_i$ 全部实例数量的比值。参与度（Participation Index）$pi(c)$ 被定义为 $\min_{i=1}^{k}\{pr(c,f_i)\}$。参与比可以用下面的方法来计算

$$pr(c,f_i)=\frac{\pi_{f_i}\left|table\_instance(c)\right|}{\left|instance(f_i)\right|},$$

其中 $\pi$ 是消除重复实例的映射操作。例如，在图 2.2 中，{A,B}的实例列为{{A.1,B.1},{A.2,B.4},{A.3,B.4}}。对特征 B 来说，有两个实例在同位模式{A,B}中，因此 $pr(\{A,B\},A)=2/5=0.4$。同样，可以算出 $pr(\{A,B\},A)$ 为 0.75。因此参与度 $pi(\{A,B\})=\min(0.75,0.4)=0.4$。

同位规则 $c_1 \Rightarrow c_2\,(p,cp)$ 的条件概率 $cp(c_1 \Rightarrow c_2)$ 是同位模式 $c_1$ 中与同位模式 $c_2$ 形成 $R$ 近邻关系实例列的比值。可以通过上述公式计算得到。在图 2.2 中，这条规则的条件概率为 $\dfrac{\pi_A \left|table\_instance(\{A,C\})\right|}{\left|table\_instance(\{A\})\right|}=2/4=50\%$。后一章对该条件概率进行了讨论。

最后把同位模式的参与度与给定的流行度阈值作比较，其参与度不小于流行度阈值的同位模式被保留，即最后发现的同位模式（这里在不引起歧义的前提下把计算中和计算后被保留的都称为同位模式）。

## 2.3 本章小结

本章对同位模式的基本概念做了详细的阐述，包括同位模式的概念、应用背景和基本算法。同位模式挖掘算法用于挖掘空间中相互接近的特征，这些特征的实例在空间中相互靠近，可以用近邻关系 $R$ 来衡量。满足近邻关系 $R$ 的实例集合可以作为同位模式的实例列，进而可以组成该同位模式的实例表。根据得到的实例表分别计算组成该同位模式的每个特征的参与比，最后计算该同位模式的参与度，与给定的流行度阈值比较后决定是否保留该同位模式。

# 第3章 基于回溯方式的时空同位模式挖掘概念

## 3.1 基于回溯方式的时空同位模式概念描述

在空间同位模式的概念之上，我们可以把同位模式的概念扩展到时空数据集中。时空数据集中的实例，具有空间坐标之外，还拥有一个时间坐标。根据实例的时间坐标，我们可以把它们整理成按时间片（Time Slot）排列，每个时间片上分布着一些特征的实例。

图 3.1 是一个**追及过程**的抽象表示：给出了一些沿时间轴分布在一维空间上的特征实例，图中纵坐标为时间，横坐标为一维空间，不同的特征分别用不同的图形表示。从图 3.1 中可以看出 A 随着时间推移，逐步向 B、C 靠近，最后特征 A 的一个实例足够接近 B、C（图中用虚线框表示“足够接近”的实例），A 的另一个实例与 B、C 一同消失在观察坐标范围之外。

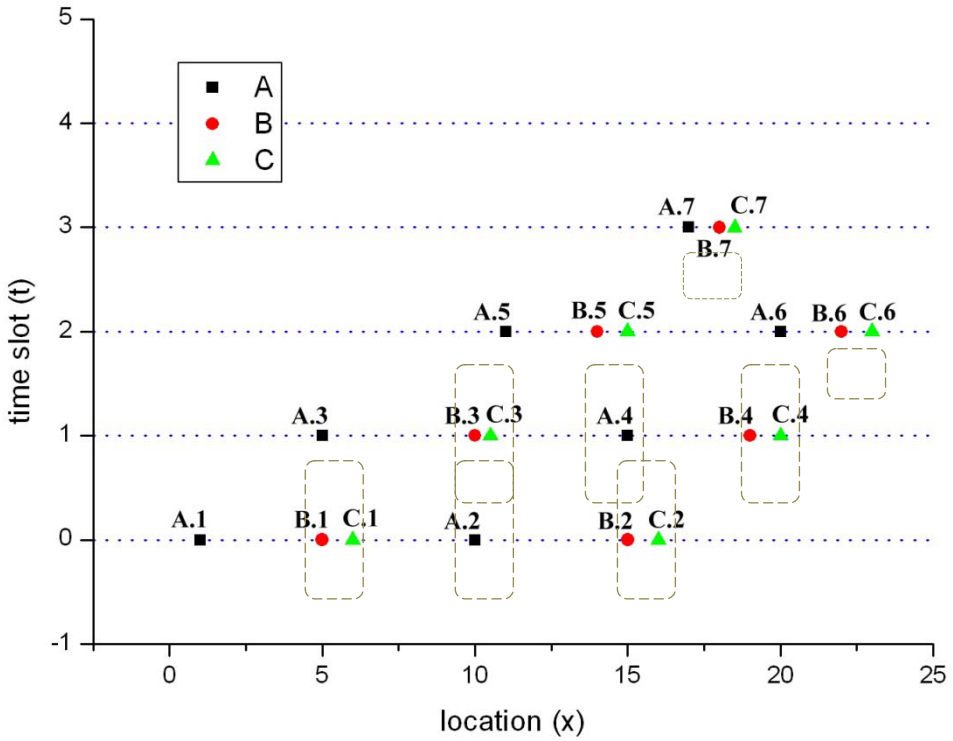



图 3.1 一维空间上的时空数据集示例

我们希望通过对这个一维时空数据集合的挖掘分析，能够发现{A，B，C}这样的时空同位模式。由于这种同位模式是跨越时间片的，与目前的时空同位模式概念有所区别，所以我们称之为基于回溯方式的时空同位模式（本文之后在不引起歧义的前提下，简称为同位模式）。

## 3.2 基于回溯方式的时空同位模式挖掘概念的形式化表述

**给出：**

**1)** **特征的集合** $F = \{f_1, f_2, ... f_k\}$；

**2)** **实例的集合** $I = \{i_1, i_2, ... i_n\}$, 每一个实例 $i_i$ 是一个向量，可以写作 $< instance\_id, feature\_type, location, time >$；

a) $instance\_id$ , 是该实例在时空数据集中的唯一标识；

b) $feature\_type \in F$ ，表明该实例所属的特征；

c) $location$ ，该实例在空间中的坐标，空间可以是用一维、二维、三维或者更高维度的坐标向量表示的；

d) $time \in N$ ，是一个整数，表明该实例在时间轴上的坐标；

**3)** **时间间距阈值**（或者称为“回溯值”）（Time Interval Threshold，$tit \in N$ ），即需要往前追溯的时间片数量，用于反映时间片之间的关联程度，由用户给出；

**4)** **权值函数** $W$（Weight Function），从另一个角度反映时间片之间的关联程度，根据应用需求，由用户给出，后面的小节会进一步说明；

**5)** **近邻关系** $R$ ，给出了实例间关系的衡量方法；

**6)** **流行度阈值**（Prevalence Threshold, $pt$），决定了一个同位模式是否流行，不流行的模式被剪枝消除；

**7)** **全局流行度阈值**（Overall Prevalence Threshold，$opt$ ）, 在整时间轴上考察一个同位模式是否流行，在全局上流行的同位模式被保留。

**目标：**

从给出的数据集中挖掘出所有的基于回溯方式的时空同位模式，这些同位模式具有下面的特性:

**1)** 计算涉及的实例与目标时间片上实例的时间间距不大于时间间距阈值

$tit$，并且有权值函数$W$体现其时间间距的影响；

**2)** 同位模式中形成实例列的实例满足近邻关系$R$；

**3)** 该同位模式在某些目标时间片上参与度不小于给定的流行度阈值$pt$；

**4)** 满足特性**3)**的同位模式在全局时间上所占的比例不低于全局流行度阈值$opt$。

## 3.3 基于回溯方式的时空同位模式建模

**定义 3.1.** 符号“·”用于表示分量，实例$i_i$的时间分量$time$，写作$i_i.time$。

图 3.1 中的 A.5, $A.5.time=2$， $A.5.feature\_type=A$。

**定义 3.2. 时间片（Time Slot）**$T_t=\{i \mid i\in I \wedge i.time=t\}$，且有$T_{t1}\bigcap T_{t2}=\varnothing$，其中$t_1\neq t_2$。

定义 3.2 的意义是，一个**无序**的时空数据集可以重新划分为**时间片的并集**$I=\bigcup T_t$。图 3.1 的时空数据集可以按时间片划分为

$I=\{A.1,A.2,B.1,B.2,C.1,C.2\}$

$\bigcup\{A.3,A.4,B.3,B.4,C.3,C.4\}\bigcup\{A.5,A.6,B.5,B.6,C.5,C.6\}\bigcup\{A.7,B.7,C.7\}$，即划分分成 4 个时间片$T_0$，$T_1$，$T_2$，$T_3$的并集。后面的章节使用记号$\bigcup T_t$表示一个经过时间片划分的时空数据集合。应用定义 3.1.的分量表示方法，可以表示一个时间片所在时间轴的位置，$T_t.time=t$。

**定义 3.3. 目标时间片（Objective Time Slot）**$T_{obj}\in\bigcup T_t$，是当时间间距阈值$tit=0$时仍然需要计算的时间片。

当时间间距阈值$tit=0$时，回溯算法退化成**混合驱动模式挖掘算法**。此时对一个固定的时间$t$，有且仅有一个时间片需要计算，即有$T_{obj}.time=t$。

**定义 3.4.** 一个时间片$T_t$对目标时间片$T_{obj}$是**有效**的，当且仅当$0\le T_{obj}.time-T_t.time\le tit$，用记号$valid(T_{obj})$表示对目标时间片$T_{obj}$有效的时间片集合，$valid(T_{obj})=\{T_t \mid 0\le T_{obj}.time-T_t.time\le tit\}$。

对目标时间片有效的时间片在时间轴上应当位于目标时间片之前（包含目标时间片），并且它们两者的时间间距不应大于时间间距阈值$tit$。仍然采用图 3.1 的数据集合$I=\bigcup T_t$，在时间间距阈值$tit=1$的条件下，对目标时间片$T_0$来说，它的有效时间片集合是$\{T_0\}$；对目标时间片$T_1$来说，它的有效时间片集合是$\{T_0,T_1\}$。之后在不特别说明的情况下，都使用时间间距阈值$tit=1$。

**定义 3.5.** 一个**合法**的**权值函数**$W$接受时间间距作为输入，输出该时间间距对应的权值，并且权值函数$W$是随时间间距增大单调递减的。

权值函数反映出与目标时间片有着不同时间间距的时间片上的实例对相应特征的影响程度，为了客观地反映距离目标时间片较远的时间片对目标时间片的影响较小的事实和使得回溯算法产生的同位模式有可以控制的参与度（定义3.6），权值函数要求具有单调递减特性。在第 6 章的实验中，我们采用的权值函数$W$形如$w^t$，其中$t$是两个时间片的时间间距，$w \le 1$是权值函数的权重因子。

**定义 3.6. 扩展的实例列（Extended Row Instance）**$row\_instance(c)$，其中的实例$i \in valid(T_{obj})$。

该定义使得实例列可以跨越几个有效时间片。图 3.1 中，对目标时间片$T_1$来说，$valid(T_1)=\{T_0,T_1\}$，计算$row_instance(\{A,B,C\})$需要考虑两个时间片，得到的实例列有$\{A.3,B.1,C.1\}$、$\{A.2,B.3,C.3\}$和$\{A.4,B.2,C.2\}$，图中用虚线框表示。

**定义 3.7. 扩展的实例表（Extended Table Instance）**$table\_instance(c)$，是所有扩展的实例列$row_instance(c)$的集合。

之后在不特别说明的情况下，实例列和实例表就是指扩展的实例列和实例表。

**定义 3.8. 命中的实例列（Hit Row Instance）**

$hit\_row\_instance(c) \in table\_instance(c) \wedge (\exists i, i \in hit\_row\_ins\tan ce(c) \wedge i \in T_{obj})$。

对目标时间片$T_1$来说，$hit_row_instance(\{B,C\})$有$\{B.3,C.3\}$和$\{B.4,C.4\}$，而实例列$\{B.2,C.2\}$没有命中，因为$B.2$、$C.2$都不属于目标时间片$T_1$。我们加入命中的实例列是希望所有有效时间片对目标时间片起到一定的影响，这些影响最后都是累加到目标时间片上的，因而没有命中目标时间片的实例列对最后产生的同位模式参与度没有贡献。

**定义 3.9. 命中的实例表（Hit Table Instance）**$hit\_table\_instance(c)$，是所有命中的实例列$hit_row_instance(c)$的集合。

**定义 3.10. 同位模式时间片（Co-location Time Slot）**

$C_{t,c,f_i}=\{i \mid i \in \pi_{f_i}(table\_instance(c)) \wedge i.time=t\}$，且有$C_{t_1,c,f_i} \bigcap C_{t_2,c,f_i}=\varnothing$，其中$t_1 \neq t_2$。

定义 3.10 是说，一个同位模式$c$在某个特征$f_i$上的实例投影可以划分成同位模式时间片的并集。再次使用图 3.1 的目标时间片$T_1$，$C_{0,\{A,B,C\},A}=\{A.2\}$，$C_{1,\{A,B,C\},A}=\{A.3,A.4\}$。

**定义 3.11. 基于时间的参与比（Participation Ratio on Time）**

$$pr_t(c, f_k) = \sum_{T_i \in valid(T_t)} \left(\frac{\left|C_{i,c,f_k}\right|}{\left|C_{i,f_k,f_k}\right|} \cdot W_{t-i}\right)。$$

为了得到基于时间的参与比 $pr_t$，需要对每一个有效的时间片计算其参与同位模式 $c$ 并且属于特征 $f_k$ 的实例占该时间片上特征 $f_k$ 全部实例的比例，接着把这个比例与权值函数相乘得到其对目标时间片的影响程度，最后把所有比值相加得到有效时间片集合对目标时间片的总体影响，即计算得到的基于时间的参与比 $pr_t$。假设权值函数 $W$ 的权重因子 $w = 0.5$，则对目标时间片 $T_1$，有

$$pr_1(\{A,B,C\}, A) = \frac{2}{2} \cdot 1 + \frac{1}{2} \cdot 0.5 = 1.25；$$

$$pr_1(\{A,B,C\}, B) = \frac{1}{2} \cdot 1 + \frac{2}{2} \cdot 0.5 = 1；$$

$$pr_1(\{A,B,C\}, C) = \frac{1}{2} \cdot 1 + \frac{2}{2} \cdot 0.5 = 1。$$

从上面可以看出，参与比 $pr_t$ 的值并不一定小于等于 1。因为所有有效时间片对目标时间片的参与比都有贡献，目标时间片上总的参与比 $pr_t$ 可能大于 1，如果权值函数是合法的（见定义 3.5），那么总的参与比是有界的 $pr_t \le tit + 1$。

对基于时间的参与比(包括马上就要提到的基于时间的参与度)，我们没有想要对它进行归一化操作，即乘以一个因子使其值小于等于 1。因为如果这样做了，就会失去各个不同时间间距阈值下产生的参与度进行相互比较的意义。即我们有一个前提，往前回溯越多的时间片，对应同一个目标时间片、同一个同位模式和特征所产生的参与比是可以进行比较的，并且是单调递增的。

**定义 3.12. 基于时间的参与度（Participation Index on Time）**

$$pi_t(c) = \min_{f_k \in c}\{pr_t(c, f_k)\}。$$

由定义 3.11 后面例子中计算出来的参与比 $pr_1(\{A,B,C\}, A)$，$pr_1(\{A,B,C\}, B)$ 和 $pr_1(\{A,B,C\}, C)$，计算参与度 $pi_1(\{A,B,C\}) = \min\{1.25, 1, 1\} = 1$。

**定义 3.13.** 某目标时间片上的一个同位模式 $c$ 是**流行的（Prevalent）**，当且仅当该同位模式基于时间的参与度不小于用户给出的流行度阈值，即 $pi_t(c) \ge pt$。

如果给出的流行度阈值是 0.5，那么同位模式 $\{A,B,C\}$ 在目标时间片 $T_0$、$T_1$、$T_2$ 和 $T_3$ 上的参与度分别是 0、1、1、1，该同位模式在目标时间片 $T_1$、$T_2$ 和 $T_3$ 上

时流行的。

**定义 3.14.** 一个同位模式 $c$ 在**全局的流行度（Overall Prevalence）**

$$op(c)=\frac{\left|\{t\mid t\in\bigcup T_{obj}.time\wedge pi_{obj}(c)\geq pt\}\right|}{\left|\{t\mid t\in\bigcup T_t.time\}\right|}。$$

定义 3.14 是说，一个同位模式在全局的流行度 $op(c)$，是若干目标时间片拥有该流行的同位模式，这些目标时间片在全局时间上所占的比值。

**定义 3.15.** 一个同位模式 $c$ **在全局时间上是流行的（Prevalent）**，当且仅当该同位模式在全局的流行度大于等于全局流行度阈值 $opt$，即

$$op(c)\geq opt。$$

接着定义 3.13 计算得到的结果，若全局流行度阈值是 0.5，那么同位模式 $\{A,B,C\}$ 在全局时间上所占比例为 $3/4=0.75\geq 0.5$，即同位模式 $\{A,B,C\}$ 在全局时间上也是流行的。

对于命中目标时间片的实例表的相关定义和计算方法，也与上述几个定义类似，这里为了建模的完整性，把命中目标时间片相关的定义列在下面，其想法与计算方式与上述的定义类似。

**定义 3.16. 命中的同位模式时间片（Hit Co-location Time Slot）**

$H_{t,c,f_i}=\{i\mid i\in\pi_{f_i}(hit\_table\_instance(c))\wedge i.time=t\}$，且有 $H_{t_1,c,f_i}\cap H_{t_2,c,f_i}=\varnothing$，其中 $t_1\neq t_2$。

**定义 3.17. 命中的参与比（Hit Participation Ratio on Time）**

$$hit\_pr_t(c,f_k)=\sum_{T_i\in valid(T_t)}\left(\frac{\left|H_{i,c,f_k}\right|}{\left|H_{i,f_k,f_k}\right|}\cdot w_{t-i}\right)。$$

**定义 3.18. 命中的参与度（Participation Index on Time）**，

$$hit\_pi_t(c)=\min_{f_k\in c}\{hit\_pr_t(c,f_k)\}。$$

**定义 3.19.** 某目标时间片上的一个同位模式 $c$ 是**命中流行的（Hit Prevalent）**，当且仅当该同位模式的命中参与度不小于用户给出的流行度阈值，即 $hit\_pi_t(c)\geq pt$。

**定义 3.20.** 一个同位模式 $c$ **在全局命中的流行度（Overall Hit Prevalence）**

$$hit\_op(c)=\frac{\left|\{t\mid t\in\bigcup T_{obj}.time\wedge hit\_pi_{obj}(c)\geq pt\}\right|}{\left|\{t\mid t\in\bigcup T_t.time\}\right|}。$$

**定义 3.21.** 一个同位模式 $c$ 在全局时间上是**命中流行的**，当且仅当该同位模

式在全局命中的流行度大于等于全局流行度阈值$opt$，即

$$hit\_op(c) \geq opt$$。

## 3.4 本章小结

本章分析了基于回溯方式的时空同位模式挖掘需求，用形式化的语言表述了这种同位模式概念，并且通过一系列的定义为基于回溯方式的时空同位模式建立了模型。

基于回溯方式的时空同位模式考察时间片之间的关联性，首先采用时间片划分输入的时空数据集，接着通过时间间距阈值$tit$、近邻关系$R$生成扩展的实例列和实例表，然后用同位模式时间片划分实例表，引入权值函数$W$来计算基于时间的参与比和参与度，最后考察一个同位模式在目标时间片上是否流行和在全局时间上是否流行。

我们还引入了命中的相关定义，以此表现有效时间片对目标时间片的影响。

# 第4章 基于回溯方式的时空同位模式挖掘算法

上一章我们为基于回溯方式的时空同位模式建立了模型，提出了时间片、扩展的实例列和实例表、同位模式时间片、权值函数、基于时间的参与比和参与度等一系列概念和定义。本章我们运用这些概念定义来构造基于回溯方式的时空同位模式挖掘算法（见算法 4.1），该算法完全基于内存。

## 4.1 算法描述

### 4.1.1 输入、输出和变量

**输入**：该算法采用的输入与我们在第 3 章基于回溯方式的时空同位模式概念的形式化表述中使用的输入一致。

1. 数据集合 $S$：为使输入数据集合简洁，我们把特征集合 $F$ 与实例集合 $I$ 合并成集合 $S$ 作为算法的输入，该集合 $S$ 包含了实例和特征的全部信息；
2. 近邻关系 $R$：是一个返回布尔值的函数，它接受两个实例作为输入，输出这两个实例是否形成近邻关系的布尔判断值；
3. 权值函数 $W$：接受时间间距作为输入，输出对应该时间间距的权值；
4. 时间间距阈值 $tit$：给定算法需要向前回溯的时间片数量；
5. 流行度阈值 $pt$：给定同位模式的流行度阈值，低于该流行度的同位模式被剪枝；
6. 全局流行度阈值 $opt$：给定同位模式在全局时间上的流行度阈值，低于该流行度的同位模式被剪枝。

**输出**：所有在全局时间上**命中流行**的基于回溯方式的时空同位模式集合。

**变量**：我们在算法中用斜体字母表示单一变量，用大写字母表示集合变量。

1. 时间片集合 $T$：是对输入的时空数据集按照时间片重新划分后的数据组织形式；
2. 时间变量 $t$：该变量每次移动都给定了一个目标时间片，算法中的后续步骤都是对该目标时间片进行计算的；
3. 同位模式大小 $k$：这个变量是同位模式挖掘算法中一直使用的一个变量，指示了目前正在挖掘的同位模式的大小，如同位模式 $\{A, B\}$ 的大小为 2，

同位模式$\{A,B,C\}$的大小为 3。因为同位模式挖掘算法具有随同位模式的大小增加单调递减的特性（证明见第 5 章），所以逐步增加挖掘的同位模式的大小可以有效减少算法的时间复杂度。

4. 同位模式候选项集合$C_{k_t}$：是包含了同位模式候选项与相应的实例表所组成的集合。

**Look-back Algorithm**

**Input**

1. S, instance set
2. R, proximity neighborhood
3. W, a weight function
4. *tit*, time interval threshold
5. *pt*, prevalence threshold
6. *opt*, overall prevalence threshold

**Output**

Sets of prevalent spatio-temporal co-location patterns overall time slots

**Variables**

1. T, re-arranged time slots
2. *t*, time_slot
3. *k*, co-location pattern size
4. C*k_t*, canditate colocations of size *k* on time_slot *t*
5. P*k_t*, prevalent colocations of size *k* on time_slot *t*
6. C*k*, all C*k_t* on whole time
7. P*k*, all P*k_t* on whole time
8. L*k*, overall hit prevalent look-back colocations of size *k*

**Algorithm**

**Step 1** T = Read_instance_generate_time_slot(S);
**Step 2** for each time_slot *t* in T {
**Step 3** C*2_t* = Gen_colocation(*t*, *tit*, T, R);
**Step 4** P*2_t* = Calculate_prune(C*2_t*, W, *pt*); }
**Step 5** L*2* = Insert_lookback_set_prune(P*2*, *opt*);

**Step 6** *k* = 2;
**Step 7** while (P*k* is not empty) {
**Step 8** for each time_slot *t* in P*k* {
**Step 9** C*k+1_t* = Gen_colocation(*t*, *tit*, P*k*, R);
**Step 10** P*k+1_t* = Calculate_prune(C*k+1_t*, W, *pt*); }
**Step 11** L*k+1* = Insert_lookback_set_prune(P*k+1*, *opt*);
**Step 12** *k* = *k* + *1*; }
**Step 13** return L;

算法 4.1 基于回溯方式的时空同位模式挖掘算法

5. 流行的同位模式集合$P_{k_t}$：是包含了通过目标时间片上流行度剪枝的同位模式和对应实例表的集合；
6. 全局同位模式候选项集合$C_k$：是在时间轴上所有同位模式候选项集合$C_{k_t}$组成的集合，$C_k = \bigcup_t C_{k\_t}$；
7. 全局流行的同位模式集合$P_k$：是在时间轴上所有流行的同位模式集合$P_{k_t}$组成的集合，$P_k = \bigcup_t P_{k\_t}$；
8. 全局命中流行的同位模式集合$L_k$：$L_k \subset P_k$并且$L_k$中的所有同位模式都是全局命中流行的。

### 4.1.2 算法步骤

我们把回溯算法分成 4 个主要部分，每个部分完成各自相对独立的计算过程：

**1. 读取输入数据集$S$并将其重新划分为时间片集合$T$**（算法步骤 1）。通常输入的数据集是一个文件，算法步骤 1 的函数把输入数据从文件读取，然后按照时间重新划分为时间片的集合$T$，$T = \cup T_t$。因为算法是基于内存的，所以时间片集合$T$保留了实例的所有信息，之后的算法步骤 3 从时间片集合$T$取得实例信息，不再访问原始数据集合。

**2. 生成大小为 2 的同位模式**（算法步骤 2、3、4、5）。因为大小为 1 的同位模式一定是流行的[1]，同位模式的计算从大小为 2 开始。回溯算法对所有的目标时间片计算同位模式（步骤 2），对于每一个目标时间片，首先计算其同位模式候选项和相应的实例表，结果存储在变量$C_{2_t}$中（步骤 3），然后根据实例表计算其参与度并与给定的流行度阈值比较后剪枝，结果存储在$P_{2_t}$中（步骤 4），最后从流行的同位模式集合$P_2$中选出全局命中流行的同位模式，存储在集合$L_2$中，对不流行的同位模式（不用满足命中条件）进行剪枝（步骤 5）。因为从时间片集合（即大小为 1 的同位模式）生成大小为 2 的同位模式的过程所采用的数据结构，与后面从$k$大小的同位模式生成$k+1$大小的同位模式（$k \geq 2$）采用的数据结构有所不同，所以在算法中把这些生成过程分成两个部分：从 1 到 2，从$k$到$k+1$。

**3. 从大小为$k$的同位模式生成大小为$k+1$的同位模式**（算法步骤 6、7、8、9、10、11、12）。该计算过程是一个循环体，循环地从大小为$k$的同位模式生成

| Time slot | 0 | 1 | | | 2 | | | 3 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Colocation | B C | A B | A C | B C | A B | A C | B C | A B | A C | B C |
| Table _instance | B1 C1<br>B2 C2 | A3 B1<br>A2 B3<br>A4 B2 | A3 C1<br>A2 C3<br>A4 C2 | B1 C1<br>B2 C2<br>B3 C3<br>B2 C2 | A5 B3<br>A4 B5<br>A6 B4 | A5 C3<br>A4 C5<br>A6 C4 | B3 C3<br>B5 C5<br>B4 C4<br>B6 C6 | A7 B7 | A7 C7 | B7 C7 |
| Prt<br>Pit | 1 1<br>1 | 1.25 1<br>1 | 1.25 1<br>1 | 1.5 1.5<br>1.5 | 1.25 1<br>1 | 1.25 1<br>1 | 1.5 1.5<br>1.5 | 1 1<br>1 | 1 1<br>1 | 1 1<br>1 |
| Hit_table _instance | B1 C1<br>B2 C2 | A3 B1<br>A2 B3<br>A4 B2 | A3 C1<br>A2 C3<br>A4 C2 | B3 C3<br>B4 C4 | A5 B3<br>A4 B5<br>A6 B4 | A5 C3<br>A4 C5<br>A6 C4 | B5 C5<br>B6 C6 | A7 B7 | A7 C7 | C7 B7 |
| Hit_prt<br>Hit_pit | 1 1<br>1 | 1.25 1<br>1 | 1.25 1<br>1 | 1 1<br>1 | 1.25 1<br>1 | 1.25 1<br>1 | 1 1<br>1 | 1 1<br>1 | 1 1<br>1 | 1 1<br>1 |

计算 $k=2$ 时同位模式实例表和参与度（步骤 3、4）

| Time slot | 0 | 1 | 2 | 3 | Overall prevalence |
|---|---|---|---|---|---|
| A B | 0 | 1 | 1 | 1 | 3/4 |
| A C | 0 | 1 | 1 | 1 | 3/4 |
| B C | 1 | 1 | 1 | 1 | 4/4 |
| A B (hit) | 0 | 1 | 1 | 1 | 3/4 |
| A C (hit) | 0 | 1 | 1 | 1 | 3/4 |
| B C (hit) | 1 | 1 | 1 | 1 | 3/4 |

$k=2$ 时计算全局流行度和剪枝（步骤 5）

| Time Slot | 1 | 2 | 3 |
|---|---|---|---|
| Colocation | A B C | A B C | A B C |
| Table _instance | A3 B1 C1<br>A2 B3 C3<br>A4 B2 C2 | A5 B3 C3<br>A4 B5 C5<br>A6 B4 C4 | A7 B7 C7 |
| Prt | 1.25 1 1 | 1.25 1 1 | 1 1 1 |
| Pit | 1 | 1 | 1 |
| Hit_Table _instance | A3 B1 C1<br>A2 B3 C3<br>A4 B2 C2 | A5 B3 C3<br>A4 B5 C5<br>A6 B4 C4 | A7 B7 C7 |
| Hit_prt | 1.25 1 1 | 1.25 1 1 | 1 1 1 |
| Hit_pit | 1 | 1 | 1 |

从 $k=2$ 的实例表生成 $k=3$ 的同位模式实例表并计算参与度（步骤 9、10）

| Time slot | 0 | 1 | 2 | 3 | Overall prevalence |
|---|---|---|---|---|---|
| A B C | 0 | 1 | 1 | 1 | 3/4 |
| A B C (hit) | 0 | 1 | 1 | 1 | 3/4 |

$k=3$ 时计算全局流行度和剪枝（步骤 11）

图 4.1 回溯算法部分计算过程

大小为 $k+1$ 的同位模式，直达不再有新的同位模式生成为止（步骤 7），并且每次循环都把变量 $k$ 加 1（步骤 12）。在循环体内部的步骤 8、9、10、11 计算过程与前一部分的步骤 2、3、4、5 相同，所使用的数据结构有所差异。

**4. 返回全局命中流行的同位模式集合** $L$（算法步骤 13）。因为全局命中流行的同位模式集合 $L_k$ 在前面的 2、3 部分已经得到了，所以这个过程比较简单，返回所有这些得到的集合 $L_k$ 就可以了。通常的做法是把集合 $L_k$ 输出到文件中，以备进一步的分析（进一步的分析在本文中我们不再考虑）。

图 4.1 是采用图 3.1 中数据集作为输入，同时设置参数：时间间距阈值 $tit=1$，权重因子 $w=0.5$，流行度阈值 $pt=0.5$，全局流行度阈值 $opt=0.5$ 的条件下回溯

算法的计算过程。

## 4.2 算法细节

上一节描述了回溯算法的整体计算过程，本节我们对算法和实现中的一些细节进行详细说明。

### 4.2.1 实例列的结构

算法采用基于连接的方法生成实例表，基于连接方法的做法是：从两个大小为 $k$ 的实例列生成一个大小为 $k+1$ 的实例列时，首先匹配大小为 $k$ 的实例列的前 $k-1$ 个实例，如果不一致，则不能生成新的实例列，如果一致，再计算两个实例列中最后的实例之间是否满足近邻关系 $R$，如果满足，则 $k+1$ 的实例列能够生成，否则不能生成新的实例列[1]。

如从大小为 2 的实例列 $\{A.3, B.1\}$、$\{A.3, C.1\}$ 生成新的实例列，首先比较分别来自两个实例列的 $A.3$ 和 $A.3$ 是否一致，一致的情况下再计算实例 $B.1$ 和 $C.1$ 是否满足近邻关系 $R$，满足了近邻关系则生成新的实例列 $\{A.3, B.1, C.1\}$。

由此我们可以知道，一个大小为 $k$ 的实例列，前 $k-1$ 个实例只是用来匹配的，不需要保留完整的实例信息，而最后一个实例是需要参与近邻关系计算的，因而保留完整的实例信息便于计算。

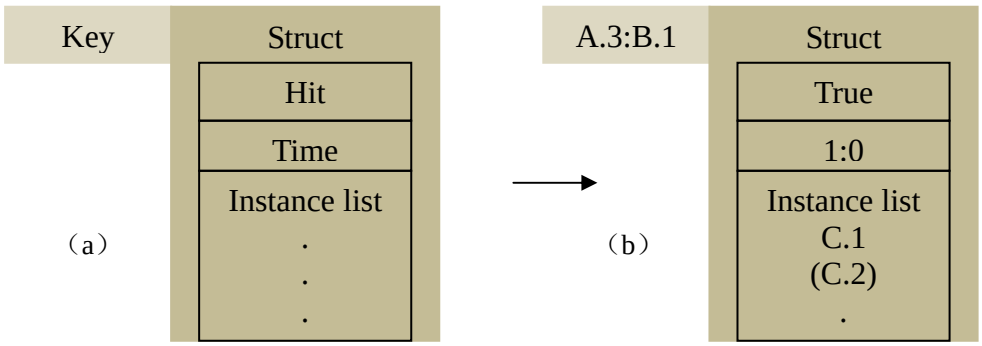



图 4.2 实例列结构

基于这个理由，我们设计了键——值（Key-Value）的实例列结构，如图 4.2 所示。在这个结构中，键存放了前 $k-1$ 个实例的简单信息（实例的 id）。值是一个结构体，这个结构体包含了该实例列的详细信息，包括该实例列的前 $k-1$ 个实例是否包含目标时间片的实例（即是否命中目标时间片），前 $k-1$ 个实例各自

所在的时间片位置等信息，最为重要的是这个结构体中包含了实例列第 $k$ 个实例的列表，列表保存了第 $k$ 个实例的全部信息。这样，通过大小为 $k$ 的实例表就可以直接生成大小为 $k+1$ 的实例表，生成过程不再需要查询初始的数据集合。

图 4.2（b）是把实例列 $\{A.3, B.1, C.1\}$ 存放在键——值结构中的一个示例，其中是否命中目标时间片和前 $k-1$ 个实例各自所在的时间片位置已经填好，该实例列中第 3 个实例 $C.1$ 存放在实例列表中。如果有另一个实例列 $\{A.3, B.1, C.2\}$ 加入，只需要在这个结构的实例列表中加入实例 $C.2$ 即可。

采用这种实例列结构可以减少存储开销，加快实例列匹配的速度。并且从大小为 $k$ 的实例列生成的大小为 $k+1$ 的实例列依然保持了这种易于存储和查询的结构，为递推计算提供了便利。

### 4.2.2 流行度剪枝

算法中有两个实例表集合 $C_{k\_t}$ 和 $P_{k\_t}$，其中 $C_{k\_t}$ 是计算实例表之后得到的，$P_{k\_t}$ 是计算流行度和剪枝之后得到的。我们在实现回溯算法时只使用了一个实例表集合，不管称之为 $C_{k\_t}$ 还是 $P_{k\_t}$。即我们在同一个实例表集合上计算实例表，计算流行度和剪枝。

前面提过回溯算法是基于内存的，因而可以节省的空间尽量要节省，并且把一个集合拷贝一份也要花费一定时间，由于这两个原因，我们把算法中提到在两个集合在实现时合并为一个。

图 4.1 回溯算法部分计算过程的步骤 3、4（步骤 9、10）给出了两个表的集合，一个是实例表，另一个是命中的实例表。为什么要使用命中的实例表，原因在定义 3.8 已经作了说明。为什么有了命中的实例表还需要保留实例表，后一章中的完备性证明会给出原因。但在实现时我们还是只使用了一个实例表，这一节我们的关注点就在于如何从这实例表同时计算参与度和命中的参与度。

从上面的实例列结构中，可以看出我们为这样的计算需求留了一条捷径：实例列结构体部分有命中和前 $k-1$ 个实例所在的时间片位置信息。我们计算参与度的时候把实例列的所有实例都划分到各自的时间片中去（可以是相应时间片数量的集合或者是若干个位图[1]），如定义 3.10 所做的那样，同时，我们也把所有命中实例列（前 $k-1$ 个实例已经命中或者第 $k$ 个实例命中）的实例划分到各自的时间片中去，这样就有了两套对应时间片的集合或者是位图——前者用于计算参与度，后者用于计算命中的参与度。

对于某个同位模式计算得到的参与度，如果小于给定的流行度阈值，则剪枝删除该同位模式和对应的实例表；但是对于计算得到的命中参与度，即使小于给定的流行度阈值，也不用剪枝删除（原理见第 5 章的完备性证明）。

### 4.2.3 全局流行度剪枝

计算全局流行度和剪枝步骤也有与上面相似的问题，即要在同一个实例表上计算全局流行度、全局命中的流行度和剪枝。

基于上一步的计算，某个同位模式在每个目标时间片上的参与度和命中的参与度都已经得到了，如图 4.1 的步骤 3、4 和步骤 9、10。我们对一个同位模式在全局时间上流行的次数和命中流行的次数分别计数，见图 4.1 的步骤 5 和步骤 11，计数结果分别除以总共的时间片数量，用这个比值与全局流行度阈值比较，判断这个同位模式在全局上是否流行和时候命中流行。对不流行的同位模式进行剪枝操作，对不是命中流行的同位模式不进行剪枝，这与上一步骤原理相同。

虽然非命中流行的同位模式不被剪枝删除，但是加入到最终命中流行的同位模式集合 $L_k$ 中的同位模式一定是全局命中流行的。

## 4.3 本章小结

本章承接第 3 章的基于回溯方式的时空同位模式挖掘概念，给出了回溯算法并对该算法进行了详细的说明和解释。

回溯算法明确了基于回溯方式的时空同位模式挖掘概念中提到的输入和输出形式，采用全部基于内存的方式来储存和处理数据。

回溯算法分为 4 个主要部分，每个部分完成各自相对独立的计算过程：

1. 读取输入数据集 $S$ 并将其重新划分为时间片集合 $T$；
2. 生成大小为 2 的同位模式；
3. 从大小为 $k$ 的同位模式生成大小为 $k+1$ 的同位模式；
4. 返回全局命中流行的同位模式集合 $L$。

之后的小节对算法的细节加以分析和说明，包括实例列的结构、流行度剪枝和全局流行度剪枝的方法。

# 第5章 基于回溯方式的时空同位模式挖掘算法分析

本章我们分析上一章提出的基于回溯方式的时空同位模式挖掘算法，重点分析算法的单调递减特性、完备性、正确性和计算复杂度。另外我们加入算法扩展一节，该小节说明如何略微改动回溯算法的参数，使其达到能够挖掘其他同位模式的目的。

## 5.1 算法单调递减特性、完备性和正确性证明

### 5.1.1 单调递减特性证明

**引理 5.1.** 对一个目标时间片，其上的基于回溯方式的时空同位模式的参与度随该同位模式大小单调递减。

证明：在一个目标时间片上的同位模式的同位模式时间片随该同位模式的大小单调递减[1]，即有 $C_{t,c,f_i},(|c|=k)\geq C_{t,c,f_i},(|c|=k+1)$。根据参与比的定义：

$$pr_t(c,f_k)=\sum_{T_i\in valid(T_t)}\left(\frac{|C_{i,c,f_k}|}{|C_{i,f_k,f_k}|}\cdot W_{t-i}\right),$$

因为 $$\frac{|C_{i,c,f_k}|}{|C_{i,f_k,f_k}|},(|c|=size)\geq\frac{|C_{i,c,f_k}|}{|C_{i,f_k,f_k}|},(|c|=size+1),$$

所以有

$$pr_t(c,f_k),(|c|=size)\geq pr_t(c,f_k),(|c|=size+1),$$

再次根据参与度的定义

$$pi_t(c)=\min_{f_k\in c}\{pr_t(c,f_k)\},$$

把上述不等式代入，有

$$pi_t(c),(|c|=size)\geq pi_t(c),(|c|=size+1),$$

即对一个目标时间片，其上的同位模式的参与度随该同位模式大小单调递减。

**引理 5.2.** 对一个目标时间片，其上的基于回溯方式的时空同位模式的全局流行度随该同位模式大小单调递减。

证明：因为对一个目标时间片，其上的基于回溯方式的时空同位模式的参与度随该同位模式大小单调递减（引理 5.1），即 $pr_t(c, f_k)$ 随同位模式 $c$ 的大小单调递减，根据全局流行度的定义

$$op(c) = \frac{\left|\{t \mid t \in \bigcup T_{obj}.time \wedge pi_{obj}(c) \geq pt\}\right|}{\left|\{t \mid t \in \bigcup T_t.time\}\right|},$$

因为 $pr_t(c, f_k)$ 是单调递减的，所以 $op(c)$ 随同位模式 $c$ 的大小单调递减。

### 5.1.2 完备性和正确性证明

**定理 5.1.** 基于回溯方式的同位模式挖掘算法是完备的。

证明：基于回溯方式的同位模式挖掘算法是完备的，当且仅当该算法发现了所有全局命中流行的同位模式，其中不考虑大小为 1 的平凡同位模式。我们通过证明算法 4.1 的所有步骤都没有遗漏任何全局命中流行的同位模式来证明该定理。

算法 4.1 中步骤 1 对输入数据按时间片进行划分，在此过程中所有实例都被保留，即有 $\bigcup T_t = S$ 。

步骤 2、8 遍历每一个时间片，即不会有时间片被遗漏。

步骤 3、9 对每个目标时间片，从大小为 $k$ 的同位模式生成大小为 $k+1$ 的同位模式。根据基于连接的同位模式挖掘算法可知，所有有效时间片上的实例都参与生成大小为 $k+1$ 的同位模式实例表，该过程没有遗漏任何实例。

步骤 4、10 对每个目标时间片上同位模式计算流行度和进行剪枝。流行度剪枝操作如算法细节所述，对非命中的同位模式不进行剪枝，被剪枝的同位模式其参与度低于给定的流行度阈值。由引理 5.1. 对一个目标时间片，其上的基于回溯方式的时空同位模式的参与度随该同位模式大小单调递减，可知被剪枝的同位模式不会影响更大的同位模式的生成，因为参与度随该同位模式大小单调递减，即使生成了更大的同位模式，其参与度一定低于给定的流行度阈值，因而还是被剪枝消除。因此该步骤不会遗漏任何流行的同位模式。

步骤 5、11 对流行的同位模式在全局时间上计算全局流行度和进行剪枝。全局流行度剪枝操作如算法细节所述，对非命中的同位模式不进行剪枝，被剪枝的同位模式全局流行度低于给定的全局流行度阈值。由引理 5.2 对一个目标时间片，其上的基于回溯方式的时空同位模式的全局流行度随该同位模式大小单调递减，可知被剪枝的同位模式不会影响更大的同位模式的生成，因为全局流行

度随该同位模式大小单调递减，即使生成了更大的同位模式，其全局流行度一定低于给定的全局流行度阈值，因而还是被剪枝消除。因此该步骤不会遗漏任何全局流行的同位模式。该步骤还得到全局流行命中的同位模式集合存储到集合 $L_k$ 中，根据算法细节的描述，所有全局命中流行的同位模式都被保留到集合 $L_k$ 中，因此该步骤不会遗漏任何全局命中流行的同位模式。

步骤 7 对所有可能大小的同位模式都进行计算，因而不会遗漏任何全局流行的同位模式。

上述步骤都不会遗漏任何全局命中流行的同位模式，因此整体算法没有遗漏任何全局命中流行的同位模式，即基于回溯方式的同位模式挖掘算法是完备的。

**定理 5.2.** 基于回溯方式的同位模式挖掘算法是正确的。

证明：基于回溯方式的同位模式挖掘算法是正确的，当且仅当被算法返回的同位模式是全局命中流行的。基于回溯方式的同位模式挖掘算法在流行度剪枝和全局流行度剪枝步骤中保证了这一性质（见算法细节）。

### 5.1.3 算法单调递减特性和完备性的意义

从引理 5.1 和引理 5.2 我们得知：对一个目标时间片，其上的基于回溯方式的时空同位模式的全局流行度随该同位模式大小单调递减。这里所说的是全局流行度，而不是全局命中的流行度。这个区别十分重要。

我们举一个例子来说明“命中”特性不是单调递减的：在目标时间片 $T_1$ 上，考虑从流行的同位模式的实例列 $\{A.1, B.1\}$ 、 $\{A.1, C.1\}$ 、 $\{B.1, C.1\}$ （其中 $A.1.time = 0$ ， $B.1.time = 1$ ， $C.1.time = 0$ ）生成更大的同位模式 $\{A, B, C\}$ 的实例列 $\{A.1, B.1, C.1\}$ ，其中 $\{A.1, B.1\}$ 是命中的， $\{A.1, C.1\}$ 不命中， $\{B.1, C.1\}$ 是命中的（这个例子在实际计算中很容易出现）。

我们采用的是基于连接的挖掘算法，因而生成实例列 $\{A.1, B.1, C.1\}$ 时，只会考虑 $\{A.1, B.1\}$ 和 $\{A.1, C.1\}$ （原理见文献[1]）。此时区别就出现了，如果我们仅考虑命中的实例列，则因为 $\{A.1, C.1\}$ 不是命中的，所以实例列 $\{A.1, B.1, C.1\}$ 无法被生成；但是如果放宽考虑范围，考虑所有流行的同位模式的实例列，因为 $\{A.1, B.1\}$ 和 $\{A.1, C.1\}$ 都是流行的同位模式的实例列，所以新的实例列 $\{A.1, B.1, C.1\}$ 可以被生成。

新的实例列 $\{A.1, B.1, C.1\}$ 包含目标时间片上 的实例 $B.1$ ，因而是命中的，即

从一个非命中的实例列$\{A.1, C.1\}$生成了命中的实例列$\{A.1, B.1, C.1\}$。这个例子说明了“命中”特性在基于连接的挖掘算法中不是单调递减的。

上面的例子很好地说明了在基于连接的挖掘算法中，流行度和命中的流行度在单调递减特性上的区别。因此我们在剪枝步骤中保留非命中的同位模式，以此来保证算法的完备性。即如果我们因为实例列$\{A.1, C.1\}$是非命中的（其对应的同位模式也很有可能是非命中的），就将其剪枝消除，在基于连接的挖掘算法中，命中的实例列$\{A.1, B.1, C.1\}$就不可能被生成，这样破坏了算法的完备性。

出现“命中”不是单调递减这个问题的原因，是因为我们的算法是基于连接的，我们必须从$\{A.1, B.1\}$与$\{A.1, C.1\}$生成$\{A.1, B.1, C.1\}$。如果有一种方式（这种方式类似 Join-less 方法[5]）可以从$\{A.1, B.1\}$、$\{A.1, C.1\}$或者$\{A.1, B.1\}$、$\{B.1, C.1\}$或者$\{B.1, C.1\}$、$\{A.1, C.1\}$来生成$\{A.1, B.1, C.1\}$，那么“命中”的单调递减特性可以保留，因为一个大小为$k+1$的命中实例列中至少有一个实例来自于两个或两个以上的大小为$k$的实例列。

## 5.2 算法的计算复杂度分析

在分析计算复杂度之前，我们先引入几个后面需要使用的变量：

1. $f = |F|$，是特征的数量；
2. $i = |I|$，是实例的数量；
3. $t = |T|$，是经过划分后的时间片的数量；
4. $\tau$，是时间间距阈值$tit+1$，即对一个目标时间片需要计算的时间数量。

然后提出利于后面分析的一些假设：

**假设 5.1.** 所有的实例平均分布在$t$个时间片上，即每个时间片上的实例数量为$i/t$。

**假设 5.2.** 所有特征对应的实例数量相等，即每个特征对应的实例数量为$i/f$。

**假设 5.3.** 所有特征在所有时间片上都出现。

### 5.2.1 全关系情形

我们首先来分析一个极端的情形。一个数据集合中的实例形成全关系，是指该数据集合中的所有实例两两之间都满足近邻关系$R$，即所有可能的不同特征下的实例组合都会成为实例列。

文献[5]对基于连接的同位模式挖掘算法的计算复杂度进行分析时指出，占用最多计算量的是从大小为$k$的实例表生成大小为$k+1$的实例表的过程。我们对这个过程的计算究竟有多么复杂的兴趣促使了本节对极端的全关系情形的分析。

首先我们来看一个最为简单的情形：只有一个时间片，即$t=1$。该时间片上每个特征对应的实例数量为$i/f$。在全关系下生成的大小为 2 的同位模式数量为$C_f^2$；从大小为 1 的同位模式到大小为 2 的同位模式需要进行的连接操作次数为（也是生成的所有实例列的数量）：

$$O(C_f^2(\frac{i}{f})^2),$$

每个大小为 2 的同位模式对应的实例列数量为$(\frac{i}{f})^2$。

然后从大小为 2 的同位模式生成大小为 3 的同位模式，生成的同位模式数量为$C_f^3$；需要执行的连接次数为

$$O(C_f^3((\frac{i}{f})^2)^2)=O(C_f^3(\frac{i}{f})^4),$$

每个大小为 3 的同位模式对应的实例列数量为$(\frac{i}{f})^4$。

由此可以推导出从大小为$k-1$的同位模式生成大小为$k$的同位模式需要的连接次数为

$$O(C_f^k(\frac{i}{f})^{2^{k-1}}) \text{（公式 5.1）},$$

因此总的时间为

$$O(\sum_{k=2}^{f} C_f^k(\frac{i}{f})^{2^{k-1}}) \text{（公式 5.2）。}$$

接下来我们去掉只有一个时间片的假设，把公式（5.2）对时间片进行扩展，并且使用变量$\tau$后有

$$O(\sum_{k=2}^{f} t\cdot C_f^k(\frac{i\cdot\tau}{f\cdot t})^{2^{k-1}}) \text{（公式 5.3）。}$$

与公式（5.2）相比，因为有$\tau\le t$，因而有

$$\tau^{2^{k-1}} / t^{2^{k-1}+1} \le 1,$$

所以公式（5.3）的计算复杂度降低了，得出的一个结论是：**对数据集合划分时间片有利于计算**。

### 5.2.2 一般情形

公式 5.2 和公式 5.3 得出的指数计算复杂度结果是难以接受的。不过这是一种全关系的极端情形。通过计算实际数据集合，我们发现在大小为 2 和大小为 3 的同位模式上生成的实例列最多。因此对于一般情况，我们可以认为生成大小为 2 的同位模式所需的连接次数

$$O(t \cdot C_f^2 (\frac{i \cdot \tau}{f \cdot t})^2) = O(\frac{i^2 \tau^2}{2t}) \quad \text{（公式 5.4），}$$

或者从大小为 2 的同位模式生成大小为 3 的同位模式所需的连接次数

$$O(t \cdot C_f^3 (\frac{i \cdot \tau}{f \cdot t})^4) = O(\frac{i^4 \tau^4}{6 f \cdot t^3}) \quad \text{（公式 5.5）}$$

是合理的计算复杂度。

当然，这只是在假设前提下的一个近似。根据实际数据的差异，算法的计算复杂度会有变化，公式（5.4）和（5.5）可以作为参考。影响计算复杂度的因素在公式给出之外还有很多，这里不一一分析了，在第 6 章的实验部分会对这些因素的影响进行实验分析。

## 5.3 算法扩展

本节说明如何改动回溯算法，使之能够用于其他的同位模式挖掘。我们主要考虑两种同位模式，一种是混合驱动模式[7]，另一种是把空间同位模式[1]应用到时空同位模式挖掘上的扩展方法。

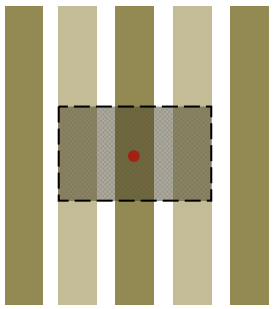
### 5.3.1 扩展回溯算法为混合驱动模式挖掘算法

这一扩展是很简便和有效的，回溯算法的最大特点是拥有时间间距阈值（回溯值）。当我们把时间间距阈值 $tit$ 设为 0 时，回溯算法就成为混合驱动模式挖掘算法。可以说混合驱动模式挖掘算法是回溯算法在 $tit = 0$ 时的特例。

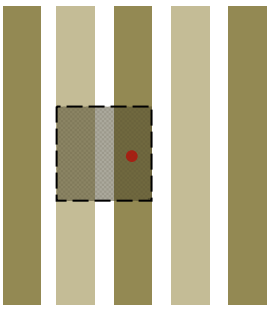
### 5.3.2 扩展回溯算法用于从空间同位模式到时空同位模式的挖掘

文献[1]提出了空间同位模式概念，并且说明了可以扩展空间坐标到时间这一维度，即把时间坐标也作为 *location* 坐标向量的一个分量，算法在使用近邻关系 *R* 考察实例之间的关系时，也把时间作为距离的因子计算进去，不同的计算方式，得到不同的效果。如图 5.1 所示，时间片用几个矩形条表示，中心点表示一个实例，虚线框包起来的区域是该实例点应用近邻关系 *R* 可以产生近邻关系的范围。

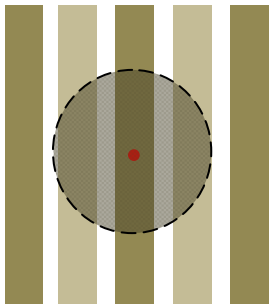
我们采用（b）方式对空间同位模式进行时空扩展，并且应用了权值概念（见定义 3.5），由此得到一个实例的近邻关系区域是近似梯形的，如（d）所示，经过这样扩展的算法我们称为算法 5.1。



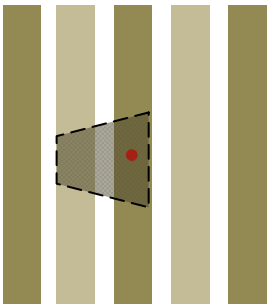
（a）两个实例的时间分量距离不大于阈值



（b）在（a）的条件下，只朝一个方向匹配



（c）时间分量作为欧式距离的一部分进行计算



（d）在（b）的条件下加入权值

图 5.1 不同方式处理时间分量示例

上一章的计算复杂度分析得出的结论：**对数据集合划分时间片有利于计算。**因此我们改进算法 5.1，把时间维度从 *location* 坐标向量中提取出来，作为实例的一个单独分量——时间分量，并以此来划分时间片。我们可以使用回溯算法对数据集合划分时间片，并且使用权值函数进行后续的计算，该算法称为**直接**

**扩展算法（Straight Extend Algorithm）**。如果原始的时间分量具有较小的度量值（小数），那么在使用回溯算法之前，我们需要对时间分量进行取整，一种方法是把所有的时间和时间阈值扩大一定的倍数，使它们都是整数；另一种方法是把一定范围内的时间分量都取整，如截断取整 $[3.0, 4.0) \approx 4$ 或者四舍五入 $[3.5, 4.5) \approx 4$，这种“取整”处理过细的时间分量方式非常实用。

## 5.4 本章小结

本章对回溯算法进行了全面的分析，包括：

1. 分析和证明算法单调递减特性、完备性、正确性；

2. 从全关系的极端情形出发，详细分析算法的计算复杂度，并且说明了一般情形的计算复杂度以及划分时间片的作用；

3. 扩展回溯算法，使之能够用于混合驱动模式的挖掘和从空间同位模式扩展的时空同位模式挖掘。

# 第6章　基于回溯方式的时空同位模式挖掘算法实验分析

本章对回溯算法行进实验分析。实验分别采用了合成数据和真实数据考察回溯算法的挖掘效果和执行效率，并与混合驱动模式挖掘算法和直接扩展算法进行比较（两种算法的描述见第 5 章的算法扩展）。

我们实现采用 C++语言实现这 3 种算法，3 种算法都是基于内存的。算法都在同一实验环境中运行，实验环境为：

CPU：Intel(R) Core(TM)2 Duo CPU E8400 @ 3.00GHz

内存：2.0GB

程序编译参数：kdevelop optimized

操作系统：Linux 2.6.26-1-686(debian 2.6.26-11)

## 6.1 合成数据生成方法

### 6.1.1 用于回溯方式和混合驱动模式的数据合成方式

我们基于文献[1]的合成数据生成方式来生成合成数据。文献[1]的数据合成分为 3 个步骤，下面对 3 个步骤作简单描述：

1. 生成同位模式候选项；
2. 对每个同位模式候选项生成实例列并把实例分散空间坐标中；
3. 在空间坐标中加入作为噪音的实例。

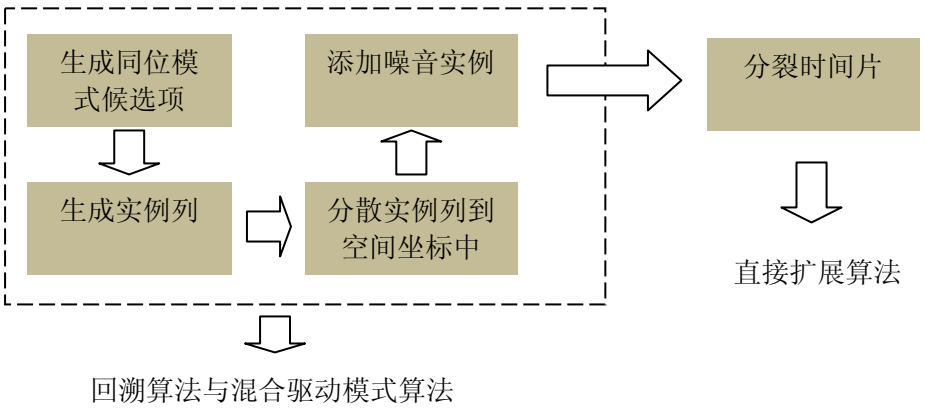



图 6.1 合成数据生成方法

表 6.1 用于合成数据的参数

| Variable | Description |
|---|---|
| *t_max* | 生成的时间片数量 |
| *co_number* | 生成的同位模式数量 |
| *co _ size* | 每个同位模式的平均大小 |
| *feature* | 所有特征的数量，缺省值为 20 |
| *frame _ x* | 二维空间坐标 x 轴范围为[0, Frame_x]，缺省值为 1000 |
| *frame _ y* | 二维空间坐标 x 轴范围为[0, Frame_y]，缺省值为 1000 |
| *distance* | 空间距离的阈值，在近邻关系 *R* 中使用，缺省值为 10 |
| *table_instance* | 每个同位模式的平均实例列数量 |
| *instance _ number* | 每个时间片上的实例数量 |
| *cross* | 时间片之间的交叉程度 |
| *slots* | 交叉的实例列中实例分散到的时间片数量，缺省值为 2 |
| *split* | 一个时间片再次分裂成的时间片数量 |

我们的合成数据生成方式对上述过程进行时间片的扩展，具体分为以下 4 个步骤（见图 6.1）：

1. 生成同位模式候选项；
2. 对每一个时间片，生成同位模式候选项对应的实例列；
3. 根据时间片的交叉程度（Cross），把实例列中的实例分散到空间坐标中；
4. 在每个时间片的空间坐标中加入作为噪音的实例。

表 6.1 给出了合成数据时用到的变量，我们对合成过程的主要步骤进行说明：

**步骤 1** 生成同位模式候选项与文献[1]一致，需要使用的变量有 *co_number*，*co _ size* 和 *feature*。

**步骤 2** 对每一个时间片都采用文献[1]的方法生成实例列，所有时间片都采用一致的同位模式候选项（由步骤 1 生成）。该步骤使用的变量有 *t_max* 和 *table_instance*。

**步骤 3** 根据参数交叉程度 *cross*（交叉程度是一个[0,1]之间的值）来确定需

要分散到别的时间片上实例列数量。交叉程度是一个分散到其他时间片的实例列占全部实例列的比例，即有 $table\_instance \cdot cross$ 数量的实例列需要随机分散到 $slots$ 个不同的时间片中，这些时间片是连续的并且位于目标时间片之前（包括目标时间片）。如在时间片 $T_1$ 上产生的一个实例列 $\{A.1, B.1\}$ 需要分散到 2 个时间片上，则分散后可能会有实例 $A.1.time = 0$ 和 $B.1.time = 1$。分散实例列仅对实例的时间分量产生影响，不改变实例的空间坐标。该步骤使用的变量 $frame\_x$、$frame\_y$、$distance$ 用于生成实例的空间坐标，变量 $cross$、$slots$ 用于生成实例的时间分量。

**步骤 4** 在每个时间片上添加噪音实例。添加的噪音数量为总的实例数量 $instance\_number$ 减去已经在实例列中使用的实例数量。并且作为噪音的实例总是生成在该时间片上的。

### 6.1.2 用于直接扩展算法的数据合成方式

我们采用上面的数据合成方式产生了可以用于回溯算法和混合驱动模式挖掘算法的数据。要把该数据用于直接扩展算法，需要经过简单的修改。

直接扩展算法与回溯算法比较，可以允许更细的时间片划分，以此体现目标时间片移动时，时间的饿连续性。

我们可以把步骤 1-4 生成的数据中的每一个时间片再次分裂成若干个时间片，引入一个分裂参数 $split$，把一个时间片上的每个实例随机分配到 $split$ 个时间片上。如时间片 $T_0$ 上的实例 $A.1$ 和 $B.1$ 在 $split = 2$ 的条件下，需要分配到新的时间片 $T_0$ 和 $T_1$ 上，可能的情况是 $A.1 \in T_0$ 和 $B.1 \in T_1$。

分裂后新的时间片数量为原来的 $split$ 倍，我们对使用分裂后的数据集合作为直接扩展算法的输入时，为了与回溯算法和混合驱动模式挖掘算法进行比较，对直接扩展算法的输入全局流行度阈值要进行修正，我们的做法是把原来的全局流行度除以分裂的数量 $split$。这样做的理由是原本在目标时间片发现的同位模式，在输入数据的时间片分裂后，还是只在一个目标时间片发现，但是此时时间片的总数是原来的 $split$ 倍，因而全局流行度阈值需要相应地缩小 $split$ 倍。

## 6.2 合成数据实验及分析

### 6.2.1 不同算法挖掘效果的实验及分析

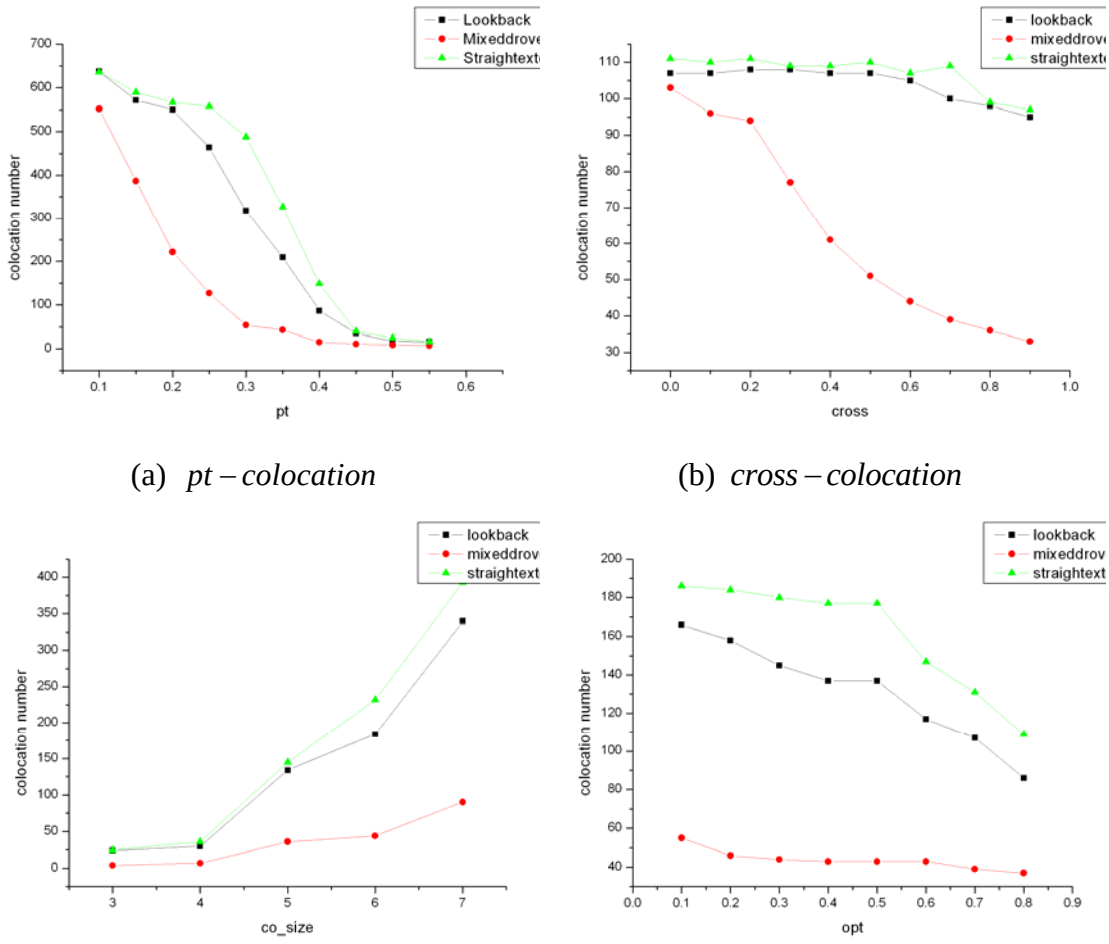
本节的实验分别考察了回溯算法、混合驱动模式挖掘算法、直接扩展算法在不同数据环境和不同算法阈值的条件下挖掘时空同位模式的效果。我们采用最终在全局命中流行的同位模式数量作为算法效果的衡量标准。

表 6.2 给出了数据合成和实验中算法采用的参数，没有列出的参数沿用缺省值。实验中三个算法都采用距离阈值 10 作为基于欧氏距离的近邻关系 $R$ 的判断标准，权值函数 $W$ 采用的权重因子为 0.5（后面的实验在不说明的情况下沿用这两个参数的值）。三个算法都采用相同的流行度阈值 $pt$ 。

表 6.2 不同算法挖掘效果的实验参数

| | Parameter | *pt* (a) | *cross* (b) | *co _ size* (c) | *opt* (d) |
|---|---|---|---|---|---|
| Generating Process | *t_max* | 10 | 10 | 10 | 10 |
| | *co_number* | 5 | 5 | 5 | 5 |
| | *co _ size* | 4 | 4 | * | 4 |
| | *table_instance* | 150 | 100 | 50 | 100 |
| | *instance _ number* | 5000 | 5000 | 5000 | 5000 |
| | *cross* | 0.5 | * | 0.5 | 0.5 |
| | *split* | 4 | 4 | 4 | 4 |
| | *pt* | * | 0.2 | 0.2 | 0.2 |
| Look-back | *tit* | 1 | 1 | 1 | 1 |
| | *opt* | 0.5 | 0.5 | 0.5 | * |
| Mixed-drove | *opt* | 0.5 | 0.5 | 0.5 | * |
| Straight-extend | *tit* | 7 | 7 | 7 | 7 |
| | *opt* | 0.125 | 0.125 | 0.125 | */4 |

考察算法效果的实验得到结果的纵坐标都是同位模式的数量，横坐标根据实验考察的算法特性有所不同，实验考察的算法特性在表 6.2 中标出，如实验 1 考察的是算法发现的同位模式数量随流行度阈值 *pt* 变化的情况，自变量 *pt* 用* 表示。




(a) $pt-colocation$ (b) $cross-colocation$

(c) $co\_size-colocation$ (d) $opt-colocation$

图 6.2 不同参数对算法挖掘效果的影响 (a) 流行度阈值 (b) 交叉程度 (c) 同位模式大小 (d) 全局流行度阈值

基于图 6.2 的实验结果，我们来分析各个因素变化对算法挖掘效果的影响。

**1. 流行度阈值 *pt* 对算法效果的影响。**三种算法随着流行度阈值升高，发现的同位模式都在减少。不流行的同位模式被剪枝删除，从同位模式挖掘算法的单调递减特性可以知道，一个被剪枝的同位模式被剪枝之后，由该同位模式作为子集的更大的同位模式都无法生成，因此随流行度阈值升高，算法发现的同位模式会迅速减少。从（a）我们还可以发现，回溯算法和直接扩展算法比起混合驱动模式挖掘算法能够发现更多的同位模式，这是前两种算法优势的体现。

**2. 交叉程度 *cross* 对算法效果的影响。**实验（b）说明了在什么情况下回溯算法比混合驱动模式挖掘算法在发现同位模式数量上有优势。*cross* 是时间片交

叉程度的反应。从（b）我们可以看到，随着 *cross* 值增大，不采用回溯方式的混合驱动模式挖掘算法能够发现的同位模式越来越少，当 $cross = 1.0$ 时，发现的同位模式数量只有交叉程度为 0 时的 $1/3$。而采用回溯方式的回溯算法和直接扩展算法发现的同位模式在时间片交叉程度变大的过程中发现的同位模式几乎没有减少。采用回溯方式的算法能够把 *cross* 这一因素对算法效果的影响降到很低的程度。

**3. 同位模式大小 *co_size* 对算法效果的影响。**随着同位模式大小的增加，该同位模式的子集数量是指数增长的，从（c）可以看出这一趋势。（c）还说明了回溯算法和直接扩展算法在同位模式大小增长时，发现的同位模式数量能够符合理论的增长数量，而混合驱动模式受制于孤立时间片的计算，无法发现与理论的增长数量一致的同位模式。

**4. 全局流行度阈值 *opt* 对算法效果的影响。**随着该值的增大，三种算法发现的同位模式逐渐减少。因为我们在全局时间上使用的是同一组同位模式（见 6.1 节合成数据生成方法），因此全局流行度阈值 *opt* 对算法挖掘得到的同位模式数量影响没有流行度阈值 *pt* 变化对算法发现的同位模式数量的影响强烈。从（d）看出当全局流行度阈值 *opt* 从 0.1 上升到 0.8 时，算法发现的同位模式数量减少了一半。同时，实验（d）说明回溯算法和直接扩展算法在发现同位模式数量上比混合驱动模式多将近 2 倍。

### 6.2.2 不同算法执行效率的实验及分析

本节的实验分别考察了回溯算法、混合驱动模式挖掘算法、直接扩展算法在不同数据环境和不同算法阈值的条件下挖掘时空同位模式的效率。我们采用算法实际的运行时间作为算法效率的衡量标准。

表 6.3 给出了数据合成和实验中算法采用的参数，没有列出的参数沿用缺省值。考察算法效率的实验得到结果的纵坐标都是算法的执行时间，横坐标根据实验考察的算法特性有所不同，实验考察的算法特性在表 6.3 中标出。

表 6.3 不同算法执行效率的实验参数

| | Parameter | *pt* (a) | *cross* (b) | *co_size* (c) | *opt* (d) | *table_instance* (e) | *t_max* (f) |
|---|---|---|---|---|---|---|---|
| Generating Process | *t_max* | 10 | 10 | 10 | 10 | 10 | * |
| | *co_number* | 5 | 5 | 5 | 5 | 5 | 5 |
| | *co_size* | 4 | 4 | * | 4 | 4 | 4 |
| | *table_instance* | 150 | 100 | 50 | 100 | * | 100 |
| | *instance_numbe* | 5000 | 5000 | 5000 | 5000 | 50* | 5000 |
| | *cross* | 0.5 | * | 0.5 | 0.5 | 0.5 | 0.5 |
| | *split* | 4 | 4 | 4 | 4 | 4 | 4 |
| | *pt* | * | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 |
| Look-back | *tit* | 1 | 1 | 1 | 1 | 1 | 1 |
| | *opt* | 0.5 | 0.5 | 0.5 | * | 0.5 | 0.5 |
| Mixed-drove | *opt* | 0.5 | 0.5 | 0.5 | * | 0.5 | 0.5 |
| Straight-extend | *tit* | 7 | 7 | 7 | 7 | 7 | 7 |
| | *opt* | 0.125 | 0.125 | 0.125 | */4 | 0.125 | 0.125 |

基于图 6.3 的实验结果，我们来分析各个参数变化对算法执行效率的影响。

**1. 流行度阈值 *pt* 对算法效率的影响。**三种算法随着流行度阈值 *pt* 升高，运行时间逐渐减少。流行度升高意味着更多的同位模式候选项被剪枝删除，被删除的同位模式无法构成更大的同位模式。不论是需要生成的同位模式候选项，还是需要连接的实例列，都随着流行度阈值 *pt* 的升高而减少，因此三种算法的执行时间都降低了。

相比混合驱动模式挖掘算法，回溯算法发现了更多的同位模式（见图 6.2(a)），因而需要花费更多的计算时间。我们再次给出第 5 章的计算复杂度公式（5.4）$O(t\cdot C_f^2(\frac{i\cdot\tau}{f\cdot t})^2)=O(\frac{i^2\tau^2}{2t})$。使用该公式我们可以来解释 3 种算法在运行时间上的差异。回溯算法与混合驱动模式挖掘算法相比较，使用公式（5.4）计算算法的时间复杂度时，回溯算法对每个目标时间片，需要计算的时间片数量 $\tau=2$，混合驱动模式挖掘算法中 $\tau=1$。由公式（5.4）可以计算得到，回溯算法的计算时间约是混合驱动模式挖掘算法的 4 倍，与图 6.3(a)的实验结果相吻合。

回溯算法与直接扩展算法相比较，回溯算法中 $\tau=2$，$t=10$，直接扩展算法中 $\tau=8$，$t=10*4=40$，代入公式（5.4）可以计算得到，回溯算法的运行时间约是直接扩展算法的 $1/4$，与图 6.2(a)的实验结果相吻合。结合回溯算法的挖掘效果（图 6.1(a)），可以看出该算法在运算时间增幅不大的情况下，挖掘到了更多的同位模式。

**2. 交叉程度*cross*对算法效率的影响。**从实验结果（图 6.3(b)）可以看出，随着交叉程度的增加，回溯算法和直接扩展算法的执行时间基本没有变化，而混合驱动模式挖掘算法的执行时间略有降低。

结合算法发现的同位模式数量（图 6.2(b)），因为回溯算法与直接扩展算法采用了回溯的方式，因而受*cross*参数变化的影响很小，所以两种算法的执行时间几乎没有变化。

而混合驱动模式挖掘算法对*cross*参数比较敏感，在该参数增大的情况下发现的同位模式减少，执行时间也会下降。这个下降幅度在图 6.1(b)中不明显，是因为除了生成同位模式步骤（算法 4.1 步骤 7、8、9、10、11、12）之外，算法还必须从大小为 1 的实例列生成大小为 2 的实例列（算法 4.1 步骤 2、3、4、5），还有一些读取数据文件和输出结果等步骤，这些步骤花费的时间基本不会变化，当生成同位模式的步骤花费很少时间时，这些执行时间基本不会变化的步骤在整体运行时间中占据的比例就提高了，导致算法整体运行时间下降不明显。三种算法之间，执行时间的差异可以参考**流行度阈值*pt*对算法效率的影响**实验中的说明。

**3. 同位模式大小*co_size*对算法效率的影响。**从图 6.3(c)可以看出，随着*co_size*参数增大，3 种算法的执行时间都有所上升。

直接扩展算法与回溯算法上升较快，并且基本保持 4 倍的比例关系，这与图 6.3(a)(b) 中直接扩展算法与回溯算法的执行时间比例一致。混合驱动模式算法的执行时间，随*co_size*参数增大，小幅度上升。上升幅度不明显的原因，一方面是该算法发现的同位模式数量增幅不大（见图 6.2(c)），另一个原因同**交叉程度*cross*对算法效率的影响**的实验结果分析一致，是由于执行时间基本不会变化的步骤在整体运行时间中占据的比例较大。

**4. 全局流行度阈值*opt*对算法效率的影响。**从实验结果（图 6.3(d)）可以看出，随着全局流行度阈值*opt*增大，三种算法的执行时间基本没有变化。

对于回溯算法和直接扩展算法来说，尽管发现的同位模式随着全局流行度

阈值*opt* 增大而减少了，但是因为算法最后发现的同位模式是全局命中流行的，而非命中的同位模式不被剪枝（见第 4 章中算法细节部分），不被剪枝的同位模式仍然参与计算。而全局流行度阈值对剪枝的影响不大，因此这两种算法的执行时间几乎没有变化。混合驱动模式挖掘算法的执行时间几乎没有变化的原因与**交叉程度*cross*对算法效率的影响**的实验结果分析相似，这里不再重复说明。三种算法之间，执行时间的差异与**流行度阈值*pt*对算法效率的影响**实验中的分析一致。

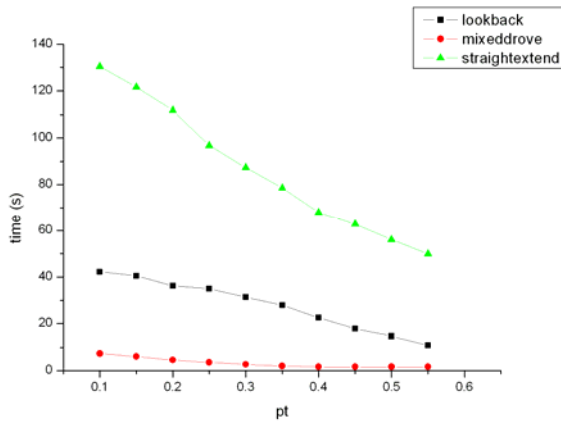
**5. 实例列数量*table _ instance*对算法效率的影响。**实例列数量对算法执行时间的影响比较复杂。首先，随着实例列数量的增加，生成同位模式时需要的连接增多，导致算法的执行时间变长，这是一个整体趋势。图 6.3(e)中，随实例列数量增加，三种算法执行时间都在增加的实验结果反映了这一整体趋势。其次，生成的实例列数量是以给定的*table _ instance* 参数为均值的泊松分布[1]，即具体的实例列数量在每次数据生成中不是固定的，因而算法的执行时间在满足整体上升趋势的情况下，会出现抖动。这在图 6.3(e)中有所体现，如对回溯算法和直接扩展算法来说，在*table _ instance* =120 位置上的执行时间高于执行时间的整体趋势线。三种算法之间，执行时间的差异与**流行度阈值*pt*对算法效率的影响**实验中的分析一致。

**6. 时间片数量*t_max*对算法效率的影响。**图 6.3(f)给出了实验结果：随着时间片数量*t_max* 增加，三种算法的执行时间呈线性增长。

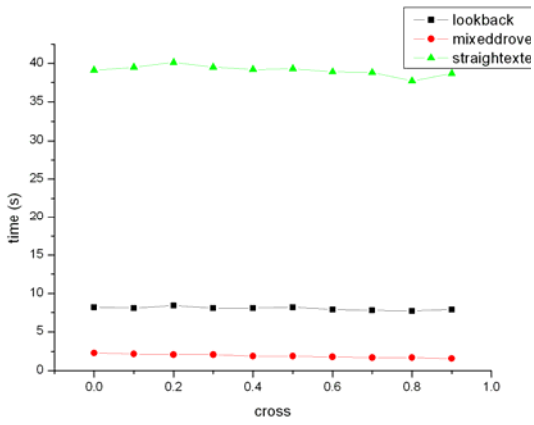
我们再次使用计算复杂度公式（5.4）$O(t\cdot C_f^2(\frac{i\cdot\tau}{f\cdot t})^2)=O(\frac{i^2\tau^2}{2t})$ 来对实验结果行进分析。当时间片数量*t_max* 改变时，全部的实例数量*i* 也改变了相同的倍数。因为每个时间片上的实例数量是固定的，为*instance _ number* ，我们把公式（5.4）中的全部的实例数量*i* 用*t_max*instance _ number* 代替，重写公式 5.4 得到

$$O(\frac{t\_max^2 instance\_number^2 t^2}{2t\_max})=O(\frac{t\_max\cdot instance\_number^2 t^2}{2})$$ （公式 5.6），

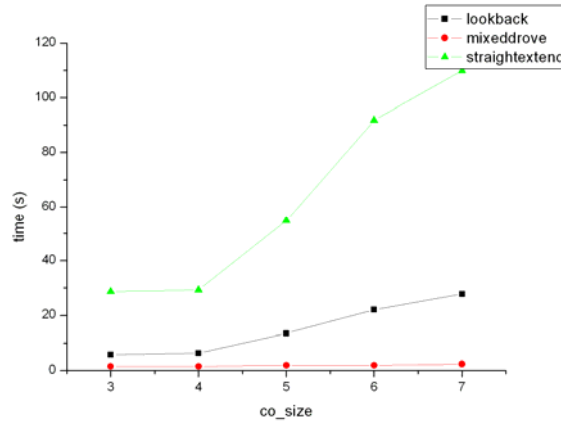
从公式（5.6）可以看出，在每个时间片上的实例数量一定的前提下，计算时间与时间片数量*t_max* 是线性关系。这与实验结果完全符合。三种算法之间，执行时间的差异与**流行度阈值*pt*对算法效率的影响**实验中的分析一致。



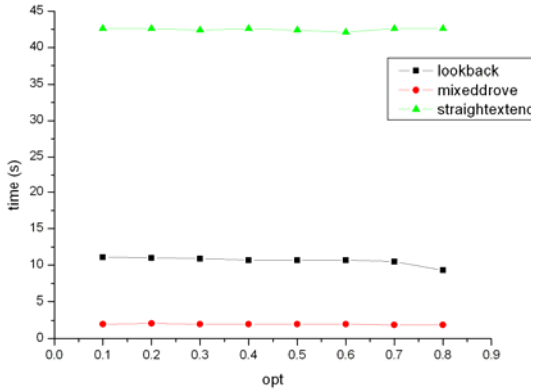

(a) $pt-time$



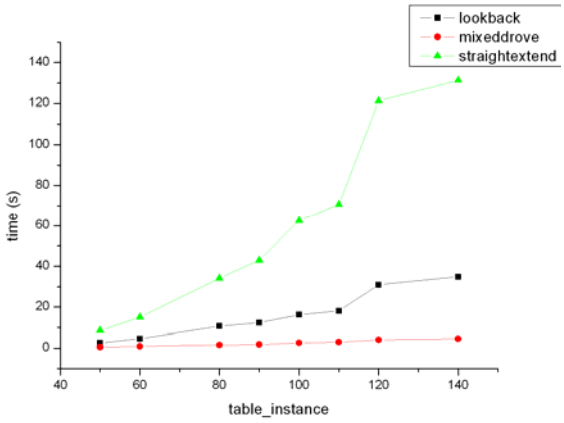

(b) $cross-time$



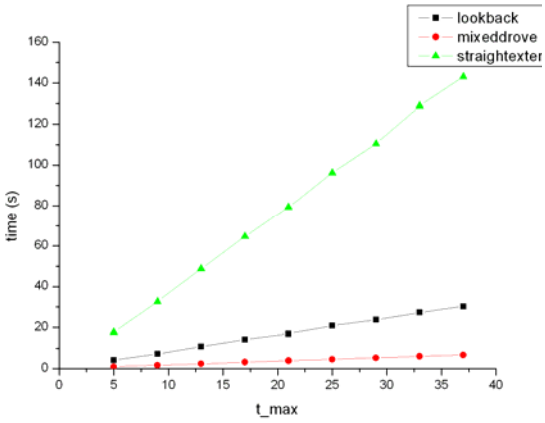

(c) $co\_size-time$




(d) $opt-time$




(e) $table\_instance-time$




(f) $t\_max-time$

图 6.3 不同参数对算法效率的影响 (a) 流行度阈值 (b)交叉程度 (c) 同位模式大小 (d) 全局流行度阈值 (e) 实例列数量 (f) 时间片数量

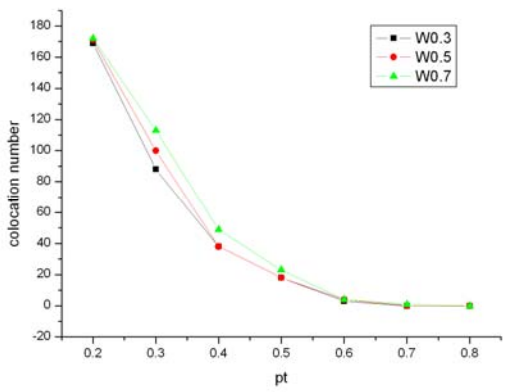
### 6.2.3 回溯算法挖掘效果和执行效率的实验及分析

本节对回溯算法的挖掘效果和执行效率作进一步的实验和分析。本节的实验考察在不同权重因子的权值函数条件下，流行度阈值、全局流行度阈值和时间间距阈值参数对算法的挖掘效果和执行效率的影响。

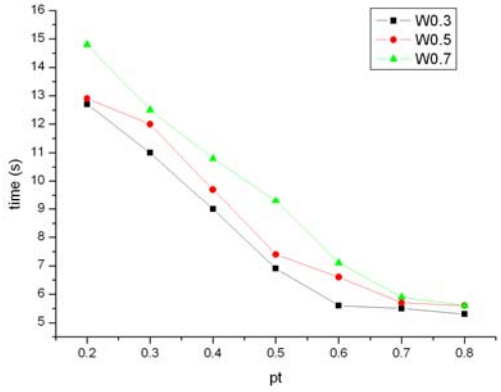
表 6.4 给出了数据合成和实验中回溯算法采用的参数，没有列出的参数沿用缺省值。实验分三组进行，每一组实验同时考查一个参数变动的条件下，在三个不同权重因子下算法的挖掘效果和执行效率。考察算法挖掘效果的实验结果纵坐标是算法发现的同位模式数量，考察算法执行效率的实验结果纵坐标是算法的执行时间。变动参数用*表示。

表 6.4 回溯算法挖掘效果和执行效率的实验参数

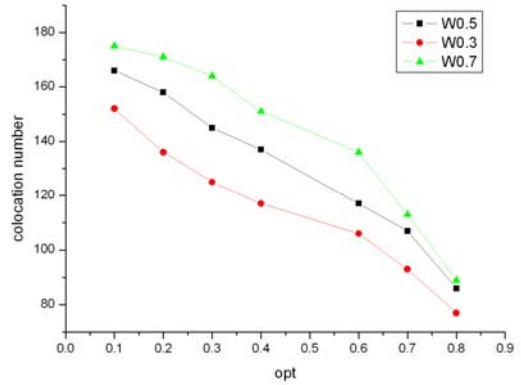
| | Variable | *pt* (a, b) | *opt* (c, d) | *tit* (e, f) |
|---|---|---|---|---|
| Generating Process | *t_max* | 10 | 10 | 10 |
| | *co_number* | 5 | 5 | 5 |
| | *co _ size* | 4 | 4 | 4 |
| | *table_instance* | 150 | 100 | 100 |
| | *instance _ number* | 5000 | 5000 | 5000 |
| | *cross* | 0.5 | 0.5 | 0.5 |
| | *split* | 2 | 2 | 5 |
| Look-back | *w* | 0.3 \| 0.5 \| 0.7 | 0.3 \| 0.5 \| 0.7 | 0.3 \| 0.5 \| 0.7 |
| | *tit* | 1 | 1 | * |
| | *pt* | * | 0.2 | 0.2 |
| | *opt* | 0.5 | * | 0.5 |



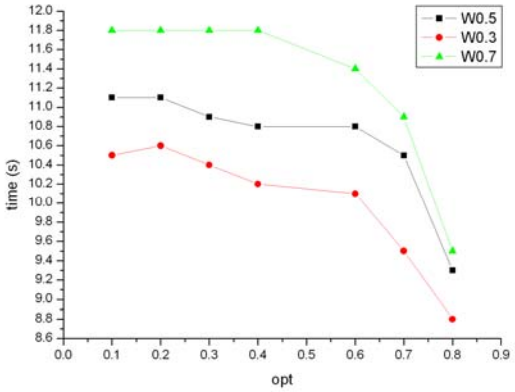

(a) $pt-colocation$



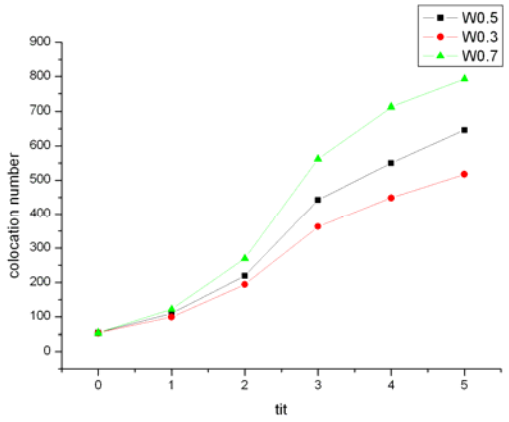

(b) $pt-time$



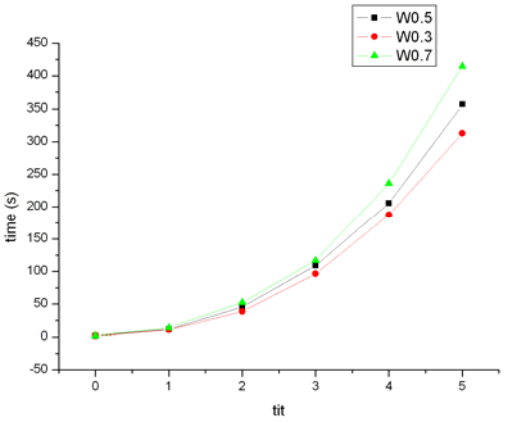

(c) $opt-colocation$




(d) $opt-time$




(e) $tit-colocation$




(f) $tit-time$

图 6.4 不同参数对回溯算法挖掘效果和执行效率的影响 (a) 流行度阈值与挖掘效果 (b) 流行度阈值与执行时间 (c) 全局流行度阈值与挖掘效果 (d) 全局流行度阈值与执行时间 (e) 时间间距阈值与挖掘效果 (f) 时间间距阈值与执行时间

基于图 6.4 的实验结果，我们来分析各个参数变化对算法挖掘效果和执行效率的影响。

**1. 流行度阈值 *pt* 对算法挖掘效果和执行效率的影响。**图 6.4(a)给出了流行度阈值对算法挖掘效果影响的实验结果，图 6.4(b)给出了流行度阈值对算法执行时间影响的实验结果。我们比较分析该参数对算法挖掘效果和执行时间的影响。

实验(a)、(b)给出的总体趋势是，随着流行度阈值增加，算法发现的同位模式和算法的执行时间都在降低，这与图 6.2(a)与图 6.3(a)的实验结果一致。算法的执行时间与发现的同位模式是正相关的，即发现更多的同位模式需要花费更多的执行时间。从图中我们可以看出，较高的权重因子使算法能发现更多的同位模式和花费较多的执行时间，但是影响并不显著。对于算法的挖掘效果，因为在目标时间片上的实例，其权值始终是 1，只有在回溯的时间片上的实例才会受到权重函数的影响，而这些实例在构成实例列的所有实例中所占的比例约为 $cross*(1/split)=0.5*0.5=0.25$，即不同的权重因子只影响了约$1/4$的实例，因此不同权值对算法挖掘效率的影响不显著。

对于算法的执行时间，一方面由于不同权重因子下发现的同位模式差距不明显，另一方面由于没有命中的同位模式不会被剪枝，即在不同权重因子下，算法需要的连接操作实际上没有减少，因而反映在实验中，在不同权重因子下算法的执行时间差距不明显。

**2. 全局流行度阈值*opt* 对算法挖掘效果和执行效率的影响。**图 6.4(c)给出了全局流行度阈值对算法挖掘效果影响的实验结果，图 6.4(d)给出了全局流行度阈值对算法执行时间影响的实验结果。我们比较分析该参数对算法挖掘效果和执行时间的影响。

从图中可以看出，随着全局流行度阈值增大，算法发现的同位模式数量和花费的运算时间都在降低。当全局流行度阈值增大时，更多的同位模式在全局时间上所占的比例低于该阈值，因而没有被发现，导致最终发现的同位模式数量减少。当同位模式的发现数量减少时，算法的执行时间首先缓慢下降，到达一个临界点之后执行时间迅速降低，见图 6.4(d)。

没有命中的同位模式不会被发现，但也不会被剪枝消除，因此算法在执行时间上的变化与挖掘效果上的变化相比是缓慢的。但是达到某一个临界值后（图 6.3(d)中 $opt=0.6$），会有大量不命中也不流行的同位模式被剪枝消除，因此算法的执行时间会很快降低。对于不用权重因子下算法挖掘效果和执行时间差异的

分析可以参考**流行度阈值 *pt* 对算法挖掘效果和执行效率的影响**的实验结果分析。

**3. 时间间距阈值*tit*对算法挖掘效果和执行时间的影响。**图 6.4(e)给出了时间间距阈值对算法挖掘效果影响的实验结果，图 6.4(f)给出了时间间距阈值对算法执行时间影响的实验结果。我们首先来分析该参数对算法挖掘效果的影响。

从图 6.4(e)可以看出，随着时间间距阈值上升，发现的同位模式数量也在增加，增加的趋势在达到一个临界点（图 6.4(e)中$tit=3$或$tit=4$）后逐渐变缓。从数据的生成方式中我们看到，所有交叉的实例列上的实例被分散到 5 个时间片上，因此理论上的实验结果是，当时间间距阈值$tit=4$时，算法可以发现所有期望被挖掘到的同位模式。实际的实验结果是，在达到$tit=4$这个点之后，算法发现的同位模式数量还在增加。

我们应该注意到基于回溯的时空同位模式中一个有意思的特性，即一个目标时间片上的特征会被有效时间片上的特征实例“点亮”。假设有一个实例列$\{A.1, B.1\}$时跨越 4 个时间片的，其中$A.1.time=3$，$B.1.time=0$，那么在计算$valid(T_3)$上的同位模式时，如果时间间距阈值$tit\le 2$，那么实例$B.1$所在的时间片不会被计算到，因而不会生成实例列$\{A.1, B.1\}$，目标时间片上的实例$A.1$对特征$A$没有贡献；如果设置时间间距阈值$tit\ge 3$，那么实例列$\{A.1, B.1\}$就能够被生成，在目标时间上的实例$A.1$就会对特征$A$产生影响。如果在目标时间片上有较多的实例列的产生方式与此类似，则会使得目标时间片上一个特征的参与此升高，进而使得某个同位模式变得流行。这个现象可以很形象地称为“点亮”。实验(e)中，当达到$tit=4$这个点之后，算法发现的同位模式数量还在增加，就是“点亮”现象的体现，即回溯更多的时间片会使得目标时间上的某些特征的参与比提高，进而产生更多的同位模式。

对于时间间距阈值对算法执行时间的影响，从图 6.4(f)可以看出。我们再次采用计算复杂度公式 5.4 $O(t\cdot C_f^2(\frac{i\cdot\tau}{f\cdot t})^2)=O(\frac{i^2\tau^2}{2t})$来帮助分析。从公式（5.4）可以明显看出，算法的计算时间与时间间距阈值（$i=tit+1$）成 2 次函数的关系，该结果与图 6.4(f)的实验结果一致。对于不用权重因子下算法挖掘效果和执行时间差异的分析可以参考**流行度阈值 *pt* 对算法挖掘效果和执行效率的影响**的实验结果分析。

## 6.3 真实数据实验

本节我们使用来自 UCI 数据库的真实数据进行实验，分别分析不同参数对回溯算法、混合驱动模式算法和直接扩展算法在挖掘效果和执行效率上的影响。

### 6.3.1 真实数据和使用方法说明

真实数据 El Nino Data（厄尔尼诺数据）来自 UCI 数据库，这是一个时空数据集。数据集包含若干可读的字段，这些字段是从分布在太平洋赤道附近的一系列浮标上得到的，该数据的作用是帮助理解和预测厄尔尼诺循环（El Nino/Southern Oscillation (ENSO) cycles）。

数据集中的每一行是某个时间点上一个浮标记录的若干字段，这些字段包括：日期、纬度、经度、东西风力（Zonal winds）、南北风力（Meridional Winds）、相对湿度、空气温度、海面温度和海面以下 500 米处的温度。一些区域中的浮标从 1980 年开始投放，数据截止到 1998 年。

我们取数据集合的南半球部分作为实验的真实数据集合，把数据中的每一行作为一个实例，并且对数据进行了如下整理，以符合实验需要：

1. 把经纬度字段投影到相对的二维地理坐标上，对该二维坐标取整（结果以米为单位）作为实例的空间坐标；
2. 把日期字段“年”作为实例的时间坐标，这样一共会产生 19 个时间片，我们删除最后一个时间片，把剩下的 18 个时间片上的实例保留；
3. 把每行之后的各个字段作二值计算后投影成一个布尔值，计算方法是，取该字段在所有行的中值作为标准，大于该中值的为布尔值“真”，小于该中值的为布尔值“假”，对于没有值的字段（一些较早投放的浮标缺少某些字段的测量值），随机给出一个布尔值；
4. 使用一个位图来存放各个字段的布尔值，每个字段对应位图的一个位（6 个字段对应 6 个位），把位图上对应字段布尔值为“真”的位置 1，否则置 0。把位图的值作为该实例的特征。因为有 6 个位，所以一共有 $2^6 = 32$ 个特征；
5. 删除数据集中重复的实例，删除重复实例的目的，是使每个特征的实例在时空坐标中是唯一的。如果一个特征中有多个实例具有同样的时空坐标，那么只要其中的一个实例形成实例列，其余的实例也要形成同样的实例列，结果是连接操作是原来的数倍，这会对算法的执行时间产生巨

大的负面影响。

步骤 5 产生的数据集作为直接扩展算法的输入数据集。对于回溯算法和混合驱动模式挖掘算法的输入数据集，我们在步骤 5 的基础上再增加两个处理步骤。

6. 把步骤 5 产生的时空数据集合上的时间片，每两个合并成一个新的时间片，即从原来的 18 个时间片合并成 9 个新的时间片；
7. 删除数据集中重复的实例。

### 6.3.2 算法挖掘效果分析

我们采用经过处理的真实数据进行算法挖掘效果的分析，重点考察回溯算法、混合驱动模式算法和直接扩展算法在流行度阈值和全局流行度阈值两个参数分别变化的条件下的挖掘效果。

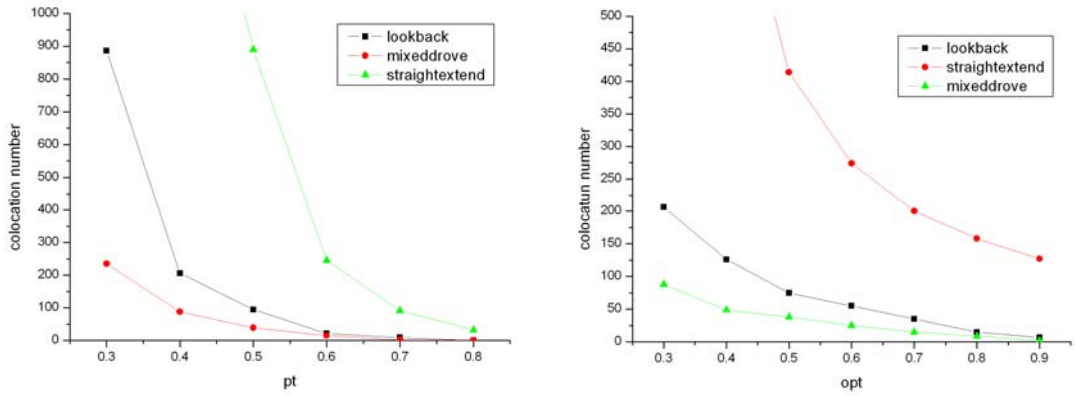
表 6.5 给出了实验中三种算法使用的参数，其中变量用*表示。算法挖掘效果的实验结果，纵坐标都是发现的命中流行的同位模式数量，横坐标根据分析的参数有所不同。

表 6.5 不同算法挖掘效果的实验参数

| | Variable | *pt* (a) | *opt* (b) |
|---|---|---|---|
| | *pt* | * | 0.4 |
| Look-back | *tit* | 1 | 1 |
| | *opt* | 0.3 | * |
| Mixed-drove | *opt* | 0.3 | * |
| Straight-extend | *tit* | 3 | 3 |
| | *opt* | 0.15 | */2 |

我们由图 6.5 的实验结果分析不同参数变化对算法挖掘效果的影响。

**1. 流行度阈值 *pt* 对算法挖掘效果的影响。**从图 6.5(a)可以看出，随着流行度阈值 *pt* 的增大，三种算法发现的同位模式数量迅速减少，这个趋势与合成数据实验的结果一致。我们从图 6.5(a)还可以发现，基于回溯的算法（回溯算法和直接扩展算法）挖掘出的同位模式比非回溯的混合驱动模式挖掘算法发现的同位模式多很多。

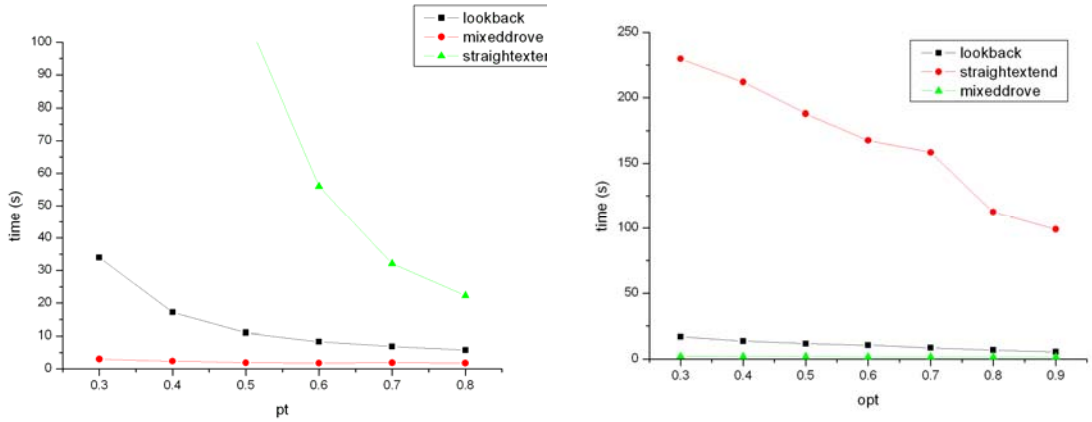



(a) *pt – colocation* (b) *opt – colocation*

图 6.5 不同参数对算法挖掘效果的影响 (a) 流行度阈值 (b) 全局流行度阈值

其中直接扩展算法发现的同位模式非常多，甚至超出了我们的预期。造成这种现象的原因，一方面是我们给回溯算法的输入数据集是合并了时间片并且删除重复实例的数据，这些被删除的实例一定会对相应的同位模式产生影响（因为这些实例的特征和空间坐标完全一样，时间坐标也至多相差 1，即它们一定形成实例列），从实验结果来看，这种影响不小；另一方面是我们假定，同位模式在全局时间上的分布是均匀的，因而对回溯算法和直接扩展算法给出了不同的全局流行度阈值（后者是前者的1/2）。这样的处理方式对合成数据比较有效。但是对于真实数据，我们并不能确切知道同位模式在全局时间上的分布情况（除非已经对该数据集进行过同位模式挖掘），当同位模式在全局时间上的分布比较集中时，合并（分裂）数据集并不能起到较好的集中（分散）实例的效果，即同位模式在全局时间上所占的比例不会因为合并（分裂）数据集而明显升高（降低），此时，对于直接扩展算法应当相应地提高全局流行度阈值，才能得到与另两种算法挖掘效果的可比结果。

**2. 全局流行度阈值*opt*对算法挖掘效果的影响。**从图 6.5(b)可以看出，随着全局流行度阈值增加，三种算法发现的同位模式数量呈下降趋势，该趋势与合成数据的实验结果一致。从图中我们可以比较地看出回溯算法发现的同位模式比混合驱动模式挖掘算法要多。对于直接扩展算法的实验结果分析，可以参考本节**流行度阈值*pt*对算法挖掘效果的影响**的实验分析。




(a) $pt-time$　　(b) $opt-time$

图 6.6 不同参数对算法执行时间的影响 (a) 流行度阈值 (b) 全局流行度阈值

### 6.3.3 算法执行效率分析

我们采用经过处理的真实数据进行算法执行效率的分析，重点考察回溯算法、混合驱动模式算法和直接扩展算法在流行度阈值和全局流行度阈值两个参数分别变化的条件下的执行效率。

表 6.5 给出了实验中三种算法使用的参数，其中变量用*表示。算法执行效率的实验结果，纵坐标都是算法的总体运行时间，横坐标根据分析的参数有所不同。

我们由图 6.6 的实验结果分析不同参数变化对算法执行时间的影响。

**1. 流行度阈值 *pt* 对算法执行时间的影响。**从图 6.6(a)可以看出，随着流行度阈值 *pt* 增大，三种算法的执行时间都在降低，这个趋势与合成数据的实验结果一致。通过与算法发现的同位模式数量（图 6.5(a)）比较，可以得知发现较多同位模式的算法花费更多的执行时间，这是因为生成更多的同位模式需要执行更多的连接操作。在流行度较低的情况下，回溯算法与混合驱动模式挖掘算法之间，直接扩展算法与回溯算法之间并没有保持理论上的执行时间的比例（理论的执行时间计算可以参考第 5 章中计算复杂度分析和上一节的合成数据实验分析），这是由于有很多流行但非命中的同位模式没有被剪枝消除。

2. 全局流行度阈值*opt* 对算法执行时间的影响。从图 6.6(b)可以看出，随着全局流行度阈值*opt* 增大，三种算法的执行时间都在降低，这个趋势与合成数据的实验结果一致。相对于混合驱动模式挖掘算法和直接扩展算法，回溯算法使

用较少的时间（与直接扩展算法比较）发现了较多的同位模式（与混合驱动模式挖掘算法比较）。混合驱动模式挖掘算法因为发现的同位模式数量较少，所以执行时间较低。对于直接扩展算法，因为发现了非常多的同位模式（参考本节**流行度阈值 *pt* 对算法挖掘效果的影响**的实验分析），因而执行时间也比其他两种算法高出很多。

## 6.4 本章小结

本章对于回溯算法的挖掘效果和执行效率，分别采用了合成数据和真实数据进行实验分析。实验考察了不同参数影响下算法的挖掘效果和执行效率，并与混合驱动模式挖掘算法和直接扩展算法进行比较。

实验部分首先说明了合成数据的生成方法，采用合成数据的实验分析得出以下结果：

1. 随着流行度阈值增大，回溯算法发现的同位模式数量与执行时间都在降低，其值在混合驱动模式挖掘算法和直接扩展算法之间，降低的幅度与被剪枝同位模式数量相关；
2. 随着全局流行度阈值增大，回溯算法发现的同位模式数量与执行时间都在降低，其值在混合驱动模式挖掘算法和直接扩展算法之间，降低的幅度与被剪枝同位模式数量相关；
3. 随着同位模式大小增加，回溯算法发现的同位模式数量与执行时间上升，其值在混合驱动模式挖掘算法和直接扩展算法之间；
4. 随着时间片交叉程度上升，回溯算法与直接扩展算法发现的同位模式数量几乎没有减少，而混合驱动模式挖掘算法发现的同位模式数量明显下降。三种算法的执行时间几乎没有随交叉程度而变化，其中回溯算法的执行时间位于混合驱动模式挖掘算法和直接扩展算法之间；
5. 随着实例列数量增加，回溯算法的执行时间上升，其值在混合驱动模式挖掘算法和直接扩展算法之间；
6. 随着时间片数量增加，回溯算法的执行时间呈线性上升，其值在混合驱动模式挖掘算法和直接扩展算法之间；
7. 不同的权重因子对回溯算法在挖掘效果和执行效率上有轻微的影响，较高的权重因子下，算法发现较多的同位模式和花费较多的执行时间。

真实数据实验部分首先对数据本身进行说明，接着整理该数据以符合实验的需要。采用真实数据的实验结果与合成数据的实验结果一致，经过分析可以得知该真实数据集的特性和对算法结果的影响。

# 第7章 总结及下一阶段研究方向

在本文中，我们考察了现有时空同位模式挖掘算法存在的问题，根据实际需求，提出了基于回溯方式的时空同位模式概念。针对一个概念，进行了形式化表述，并且建立了基于回溯方式的时空同位模式模型。基于该模型，我们给出了基于回溯方式的时空同位模式挖掘算法。该算法采用基于连接的方式，并且使用了高效的实例列结构。算法细节中我们详细讨论了流行度剪枝和全局流行度剪枝的方法。在算法分析部分，我们证明了回溯算法的单调递减特性、完备性和正确性，并对这些性质给与了充分的讨论。我们给出了算法的计算复杂度公式，使用该公式，我们很容易解释实验中产生的现象。我们指出，混合驱动模式挖掘算是是回溯算法在时间间距为 0 时的一个特例，同时，我们对空间同位模式挖掘算法进行扩展，使之能够用于挖掘时空数据。在实验部分，我们分别采用了合成数据和真实数据来进行实验。实验分析在不同参数影响下，回溯算法在挖掘效果和执行效率上的性能，并与混合驱动模式挖掘算法和直接扩展算法进行比较。

对于时空数据集的处理方式，挖掘同位模式是其中的一种。本文提出的回溯算法，在挖掘效果上已经高于目前的时空同位模式挖掘算法，但是在效率上还可以进一步提升。下一阶段的研究以改进该算法的效率为主，放弃基于连接的框架，运用 Join-less 的思想来构建算法，考虑怎样对非命中的同位模式进行剪枝来保证算法的单调递减特性、完备性和正确性。

# 参考文献

[1] Y. Huang, S. Shekhar, and H. Xiong, Discovering Co-location Patterns from Spatial Datasets: A General Approach[J]. IEEE，Trans. on Knowledge and Data Eng. (TKDE), 2004, 16(12): 1472-1485

[2] Agrawal, R, et al. Fast Algorithms for Mining Association Rules. Proc. of 20th Int. Conf. on Very Large Data Bases (VLDB), 1994.

[3] J. S. Yoo and S. Shekhar, A Partial Join Approach for Mining Co-location Patterns[J], ACM-GIS'05, Washington D.C., USA, 2005: 241-249

[4] C. Berge. Graphs and Hypergraphs. American Elsevier, 1976.

[5] J. S. Yoo and S. Shekhar, A Joinless Approach for Mining Spatial Colocation Patterns[J]. IEEE Trans. on Knowledge and Data Eng. (TKDE), 2006. 18(10): 1323-1337

[6] Jin SoungYoo, et al. Discovery of Co-evolving Spatial Event Sets. Proc. Of 6th SIAM Int. Conf. on Data Mining (SDM), 2006.

[7] Mete Celik, et al. Mixed-Drove Spatio-Temporal Co-occurrence Pattern Mining: A Summary of Results. Proc. of 6th IEEE Int. Conf. on Data Mining (ICDM), 2006 .18(22): 119-128

[8] Mining At Most Top-K% Mixed-drove Spatio-temporal Co-occurrence Patterns: A Summary of Results

[9] Jin SoungYoo, et al. Mining Time-Profiled Associations: An Extended Abstract. Proc. of Int. Pacific-Asia Conf. on Knowledge Discovery and Data Mining (PAKDD), 2005.3518: 136-142

[10] Yan Huang, et al. A Framework for Mining Sequential Patterns from Spatio-Temporal Event Datasets. TKDE 2008.

[11] Mete Celik, et al. Sustained Emerging Spatio－Temporal Co－occurrence Pattern Mining: A Summary of Results. Proc. of 18th IEEE Int. Conf. on Tools with Artificial Intelligence (ICTAI) 2006.12(19): 106-115

# 致谢

基于回溯方式的是同位模式概念和挖掘算法的提出，是我与钱烽学长多次讨论的结果，首先向他表示感谢。他对于该论文的最终成形进行了全程的耐心指导。

在实验分析部分，何江枫学长给我讲解了之前数据的合成方法，正式基于他的工作，我才得以为本文的算法设计相应的合成数据实验。

同时我要感谢同为大四的寝室室友也是同一个实验室的室友何琪，我们在完成毕业设计的过程中经常相互讨论，交流经验。正是这些讨论和交流，使我在算法实现中努力使用好的 C++语言实现方式。

最后要感谢我的导师何钦铭教授，他一直关心我毕业设计的进展情况，给我增添了信心，使我能够较好地完成毕业设计。

# 本科生毕业论文（设计）任务书

一、题目：一种基于回溯方式的时空同位模式挖掘算法

二、指导教师对毕业论文（设计）的进度安排及任务要求：

1) 考察现有的时空同位模式挖掘方法，细化基于回溯方式的时空同位模式挖掘概念：2009 年 3 月中旬至 2009 年 3 月底；
2) 提出完整的基于回溯方式的时空同位模式挖掘理论框架： 2009 年 4 月初至 2009 年 4 月中旬；
3) 为这一理论框架设计算法并编成实现：2009 年 4 月中旬至 2009 年 4 月底；
4) 用实验方法验证算法和分析其挖掘效果和计算效率：2009 年 5 月初至 2009 年 5 月中旬；
5) 整理并完成论文：2009 年 5 月中旬至 2009 年 5 月底

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