An Investigation in Efficient Spatial Patterns Mining

by

Lizhen Wang

A thesis submitted to the School of Computing and Engineering
of the University of Huddersfield
for the degree of Doctor of Philosophy

School of Computing and Engineering
The University of Huddersfield
(April 2008)

Abstract

The technical progress in computerized spatial data acquisition and storage results in the growth of vast spatial databases. Faced with large amounts of increasing spatial data, a terminal user has more difficulty in understanding them without the helpful knowledge from spatial databases. Thus, spatial data mining has been brought under the umbrella of data mining and is attracting more attention.

Spatial data mining presents challenges. Differing from usual data, spatial data includes not only positional data and attribute data, but also spatial relationships among spatial events. Further, the instances of spatial events are embedded in a continuous space and share a variety of spatial relationships, so the mining of spatial patterns demands new techniques.

In this thesis, several contributions were made. Some new techniques were proposed, i.e., fuzzy co-location mining, CPI-tree (Co-location Pattern Instance Tree), maximal co-location patterns mining, AOI-ags (Attribute-Oriented Induction based on Attributes' Generalization Sequences), and fuzzy association prediction. Three algorithms were put forward on co-location patterns mining: the fuzzy co-location mining algorithm, the CPI-tree based co-location mining algorithm (CPI-tree algorithm) and the order-clique-based maximal prevalence co-location mining algorithm (order-clique-based algorithm). An attribute-oriented induction algorithm based on attributes' generalization sequences (AOI-ags algorithm) is further given, which unified the attribute thresholds and the tuple thresholds. On the two real-world databases with time-series data, a fuzzy association prediction algorithm is designed. Also a cell-based spatial object fusion algorithm is proposed. Two fuzzy clustering methods using domain knowledge were proposed: Natural Method and Graph-Based Method, both of which were controlled by a threshold. The threshold was confirmed by polynomial regression. Finally, a prototype system on spatial co-location patterns' mining was developed, and shows the relative efficiencies of the co-location techniques proposed

The techniques presented in the thesis focus on improving the feasibility, usefulness, effectiveness, and scalability of related algorithm. In the design of fuzzy co-location

mining algorithm, a new data structure, the binary partition tree, used to improve the process of fuzzy equivalence partitioning, was proposed. A prefix-based approach to partition the prevalent event set search space into subsets, where each sub-problem can be solved in main-memory, was also presented. The scalability of CPI-tree algorithm is guaranteed since it does not require expensive spatial joins or instance joins for identifying co-location table instances. In the order-clique-based algorithm, the co-location table instances do not need be stored after computing the *Pi* value of corresponding co-location, which dramatically reduces the executive time and space of mining maximal co-locations. Some technologies, for example, partitions, equivalence partition trees, prune optimization strategies and interestingness, were used to improve the efficiency of the AOI-ags algorithm. To implement the fuzzy association prediction algorithm, the “growing window” and the proximity computation pruning were introduced to reduce both I/O and CPU costs in computing the fuzzy semantic proximity between time-series.

For new techniques and algorithms, theoretical analysis and experimental results on synthetic data sets and real-world datasets were presented and discussed in the thesis.

Acknowledgements

Though this research has been a mostly solitary effort during periods of the Doctor of Philosophy Degree, there are many people to whom I am indebted for support and assistance in various ways. Without them, this would never have been completed, and it is appropriate that they should share in it.

I would like to express my deep gratitude to Dr. Lu for her supervision and invaluable suggestions. Her invaluable comments on concepts, structures and organization have greatly enhanced any value the thesis may have. Furthermore, Dr. Lu's expectation and encouragement aroused me continue to do my best for my thesis.

I would especially like to thank Professor Yip for paying attention to my research. With his recognizing and support, my thesis could come forth in such short time. My sincere thanks are due to Mrs Lihong for her constant help in improving my English writing.

I would like to thank my husband, Zizhong, for his love, encouragement and understanding. My particular thanks are directed to my daughter, Beisi, for her love, expectations and especially the power come from her super excellence grade in Chinese admission examination.

Finally, I have to say that the happiness of success have been tasted from the processing of finishing this thesis. I will go on explore in the field of the spatial data mining.

Contents

List of Figures

List of Tables

Publications

The following papers are the results of the work conducted during Doctor of Philosophy Degree:

[1] **Wang, L**., Lu, J., Yip, J. (2007) AOG-ags Algorithms and Applications. In: Proceedings of the Third International Conference on Advanced Data Mining and Applications (ADMA 2007), Springer-Verlag, Berlin, LNAI 4632, pp. 323-334, 2007.8

[2] **Wang, L**., Lu, J., Yip, J. (2007) An Effective Approach to Predicting Plant Species in an Ecological Environment, *In:* Proceedings of the 2007 international Conference on Information and Knowledge Engineering (IKE' 07), Las Vegas Nevada, USA, June 25-28, 2007, pp. 245-250

[3] **Wang, L.**, Bao, y., Lu, J., Yip, J. (2008) A New Join-less Approach for Co-location Pattern Mining, *In:* Proceedings of the IEEE 8th International Conference on Computer and Information Technology (CIT2008), Syney, Australia 8-11 July 2008 (Accepted)

[4] **Wang, L.**, Lu, J., Yip, J. (2008) An Order-Clique-Based Approach for Mining Maximal Co-location Patterns, University of Huddersfield, Poster, 2008

The papers under review:

[5] **Wang, L.**, Lu, J., Yip, J. Discovering Co-location Patterns from Fuzzy Spatial Data Sets, Submitted to the International Journal of Information Sciences, Elsevier, December, 2007.

[6] **Wang, L.**, Zhou, L., Lu, J., Yip, J. An Efficient Approach for Mining Maximal Co-location Patterns, Submitted to The International Journal of Information Sciences, Elsevier, April, 2008.

[7] **Wang, L.**, Bao, Y., Lu, J., Yip, J. A Visual Spatial Co-location Patterns' Mining Prototype System, Submitted to International Conference on Cyberworlds (CW 2008), Hangzhou, China, Sept. 22-24, 2008, April, 2008.

Terminologies

λ ----the fuzzy equivalence partition threshold (the level value)

AGS---- Attributes' Generalization Sequence

AOI----Attribute-Oriented Induction

AOI-ags---- Attribute-Oriented Induction based on Attributes' Generalization Sequence

CDM----Co-location Data Management

CMA----Co-location Mining Analysis

CP----Conditional Probability

CP_m-tree----Candidate Maximal ordered Prevalence co-locations TREE

CPA----Co-location Patterns Applying

CPI-tree----Co-location Patterns Instance TREE

CPM----Co-location Patterns Mining

EM----Equivalence Matrix

EPC----Fuzzy equivalence Classifications

GBM----Graph-Based Method

GFCG----General Fuzzy Co-location Generation

IDE----Initial Data Exploration

IDF----Inverse Document Frequency

Ins-tree----table Instances' inspecting TREE

Min-Prev----Prevalence value Threshold

Min-Cond-Prob----Conditional Probability Threshold

MMM----Modified Matrix Method

NM----Natural Method

Neib-tree----Neighbour relationship TREE

OFCG----Optimized Fuzzy Co-location Generation

P_2-tree----Prevalence size-2 co-location header relationship TREE

PD----the Degree of Proximity between two time-series

Pi----Participation Index

Pr----Participation Ratio

S----Similarity Matrix

SCPMiner---Spatial Co-location Patterns' Mining Prototype System

SDM----Spatial Data Mining

SOLAP----Spatial On-Line Analysis Processing

SP----Semantic Proximity

Chapter 1

Introduction

Spatial data mining refers to the extraction of knowledge, spatial relationships, or other interesting patterns not explicitly stored in spatial data sets. It is expected to have wide applications in geographic information systems, geo-marketing, remote sensing, image database exploration, medical imaging, navigation, traffic control, environmental studies, and many other application areas where spatial data are used. A crucial challenge to spatial data mining is the exploration of efficient spatial data mining techniques due to the huge amount of spatial data, and the complexity of spatial data types and spatial access methods.

The final goal of the thesis is to develop some novel theoretical concepts and methods for spatial patterns mining, and develop a prototype system to explore the implementation of a spatial data mining system. To fulfil the goal, the following work will be carried out:

(1). Extend mining spatial co-location patterns from general spatial data sets for mining fuzzy spatial co-location patterns from fuzzy spatial data sets.

(2) Design a new join-less algorithm for identifying co-location pattern table instances.

(3) Present an order-clique-based method for mining maximal prevalence co-location patterns.

(4). Survey the efficiency of mining correlations between attributes based on attributed-oriented induction (AOI, for short), and expand the traditional AOI, based on attributes' generalization sequence, to upgrade.

(5). Study mining prediction technologies exhaustively, and based on the concept of semantic proximity, employ a method, evaluating the fuzzy association degree, to solve the problem of spatial mining prediction.

(6). Propose a cell-based spatial object fusion method in spatial data sets, which only uses locations of objects and without distance between two objects.

(7). Investigate fuzzy clustering methods based on domain knowledge.

(8). Explore the implementation of a visual spatial co-location patterns' mining prototype system (SCPMiner).

1.1 Motivation

In this section, the arguments of this thesis are briefly stated.

Spatial data mining has attracted a great deal of attention from not only the spatial information industry but also the whole society in recent years, due to the wide availability of huge amounts of spatial data and the imminent need for turning such spatial data into useful spatial information and knowledge. The spatial information and knowledge gained can be used for applications ranging from forestry and ecology planning, to provide public service information regarding the location of telephone and electric cables, pipes, and sewage systems.

Spatial data, like geographic (map) data, very large-scale integration (VLSI) or computed-aided design data, and medical or satellite image data contain spatial-related information. Spatial data may be represented in **raster format**, consisting of n-dimensional bit maps or pixel maps. For example, a 2-D satellite image may be represented as raster data, where each pixel registers the rainfall in a given area. Also, the data information can be represented in **vector format**, where roads, bridges, buildings, and lakes are represented as unions or overlays of basic geometric constructs, such as points, lines, polygons, and the partitions and networks formed by these components.

Spatial data can now be stored in many different kinds of spatial databases and information repositories. A spatial data repository architecture that has emerged is the **spatial data warehouse**, a repository of multiple heterogeneous data sources organized under a unified schema at a single site in order to facilitate management decision making. Spatial data warehouse technology includes spatial data cleaning, spatial data integration, and **spatial on-line analytical processing** (SOLAP), that is, analysis techniques with functionalities such as summarization, consolidation, and aggregation as well as the ability to view information from different angles. Although SOLAP tools support multidimensional analysis and decision making, additional spatial data analysis tools are required for in-depth analysis, such as spatial data classification, spatial co-location mining, and spatial outlier detection. In addition, huge volumes of spatial data can be accumulated beyond spatial databases and spatial data warehouses. How to analyse spatial data in such different forms effectively and efficiently becomes a challenge.

The abundance of data, coupled with the need for powerful data analysis tools, has been described as a *data rich but information poor* situation, especially for the spatial data field. The fast-growing, tremendous amount of spatial data, collected and stored in large and numerous spatial data repositories, has far exceeded human ability for com-

prehension because powerful tools are lacking. As a result, spatial data collected in repositories become “spatial data tombs”----spatial data archives that are seldom visited.

The tools for spatial data mining perform spatial data analysis and may uncover important spatial data patterns, contributing greatly to business strategies, ecology planning, and scientific and medical research. The widening gap between spatial data and spatial information calls for a systematic development of *spatial data mining tools* that will turn data tombs into “golden nuggets” of knowledge.

A question is raised: “What kinds of data mining methods should be performed on spatial data sets?” Mining Spatial data is supposed to uncover spatial patterns which may describe the characteristics of plants located near a specified kind of location, such as an alpine terrain, the species diversity of mountainous areas located at various altitudes, or the change in trend of metropolitan poverty rates based on city distances from major highways. That is, the spatial relationships among a set of spatial objects need to be dug out through spatial data mining in order to discover which subsets of objects are spatially auto-correlated or associated. Besides, during a mining process, clusters and outliers also need be identified by spatial cluster analysis, and spatial classification should be provided to construct models for prediction based on the relevant set of features of the spatial objects.

Data mining in spatial databases is different from that in relational databases in the sense that attributes of the neighbours of some objects of interest may have an influence on the object (Han and Kamber, 2006; Ester et al, 1999; Lee et al, 2007). The explicit location and extension of spatial objects define the implicit relations of spatial neighbourhoods (such as topological, distance and direction relations) that are used by spatial data mining algorithms [Ester et al, 1998; Ester et al, 1999; Kriegel et al, 2004]. Therefore, the crucial challenge in spatial data mining is the efficiency of spatial data mining algorithms and the effective application of spatial data mining technology, due to the huge amount of spatial data, and the complexity of spatial data types and spatial methods.

1.2 Background in Spatial Data Mining

Following the discussion of the demand on spatial data mining, and the importance of its applications, we take a further look at the research work, done so for in this field. In fact, spatial data mining has been studied extensively. A comprehensive survey on spatial data mining methods can be found in the papers of Ester, Kriegel, Sander (1997) and Shekhar and Chawla (2003). Lu, Han and Ooi (1993) proposed a generalization-based spatial data mining method by attribute-oriented induction. Ng and Han (1994) proposed

performing descriptive spatial data analysis based on clustering results instead of on predefined concept hierarchies. Zhou, Truffet, and Han (1999) proposed efficient polygon amalgamation methods for on-line multidimensional spatial analysis and spatial data mining. Koperski and Han (1995) proposed a progressive refinement method for mining spatial association rules. Spatial classification and trend analysis methods have been developed by Ester, Kriegel, Sander, and Xu (1997); and Ester, Frommelt, Kriegel, and Sander (1998). A two-step method for classification of spatial data was proposed by Koperski, Han, and Stefanovic (1998). A spatial data mining system prototype, Geominer, was developed by Han, Koperski, and Stefanovic [HKS97].

For further background of spatial data mining techniques explored in the thesis is discussed in detail below.

1.2.1 Spatial Co-location Pattern Mining

Related approaches for discovering spatial co-location patterns can be classified into two categories, in the literature, spatial statistics-based mining and data mining approaches.

（1） Spatial statistics-based mining approaches use measures of spatial correlation to characterize the relationship between different types of spatial features. Measures of spatial correlation include the cross-K function with Monte Carlo simulation and mean nearest-neighbour distance proposed by Cressie (1991), and spatial regression models was proposed by Chou (1997). Computing spatial correlation measures for all possible co-location patterns can be computationally expensive due to the exponential number of candidate subsets given a large collection of spatial Boolean features.

（2） Data mining approaches can be further classified into clustering-based map overlay approaches and association rule-based approaches.

a. A clustering-based map overlay approach treats every spatial attribute as a map layer and considers spatial clusters (regions) of point-data in each layer as candidates for mining associations that was proposed by Estivil-Castro and Lee, (2001); and Estivil-Castro and Murray, (1998). Given X and Y as sets of layers, a clustered spatial association rule is defined as $X \Rightarrow Y(cs, cc\%)$, for $X \cap Y = \Phi$, where *cs* is the clustered support, defined as the ratio of the area of the cluster (region) that satisfies both X and Y to the total area of the study region S, and cc% is the clustered confidence, which can be interpreted as cc% of areas of clusters (regions) of X intersect with areas of clusters (regions) of Y. The complexity and the efficiency are the crucial problems in the clustering-

based approaches.

b. Association rule-based approaches can be further classified into transaction-based approaches and distance-based approaches.

- Transaction-based approaches focus on defining transactions over space. For example, Koperski and Han (1995) and Wang et al (2005) proposed transactions over space defined by a reference-object centric model. Under this model, transactions are created around instances of one user-specified spatial object. The spatial association rules are derived using the Apriori (Agarwal and Srikant, 1994) algorithm. However, it is nontrivial to generalize this paradigm to the case where no reference feature is specified. Also, defining transactions around locations of instances of all features may yield to duplicate counts for many candidate associations.

- The distance-based approach was first presented by Shekhar and Huang (2001) and Morimoto (2001). Related work may be classified into three categories, which are the join-based approach (Shekhar and Xiong, 2001; Xiong et al, 2004; Huang et al, 2004), the partial join approach (Yoo and Shekhar, 2004) and the join-less approach (Yoo et al, 2005). The instance join-based co-location mining algorithm is similar to Apriori (Agarwal and Srikant, 1994). First, after finding all neighbour pair objects (size 2 co-location instances) using a geometric method, the method finds the instances of size k (>2) co-locations by joining the instances of the size k-1 subset co-locations where the first k-2 objects are common. This approach finds correct and complete co-location instance sets. However, the join-based approach is computationally expensive with the increase of co-location patterns and their instances. The partial join approach converts a continuous spatial data into a set of disjoint clique neighbourhoods while keeping track of the spatial neighbour relations not modelled by the transactionization. This approach reduces the number of expensive join operations dramatically in finding co-location instances. However, the performance depends on the distribution of the spatial dataset, especially the number of cut neighbour relations. Yoo, Shekhar and Celik (2005) proposes a novel join-less approach for co-location pattern mining, which materializes spatial neighbour relationships with no loss of co-location instances and reduces the computational cost of identifying the instances. The join-less co-location mining algorithm is efficient since it uses an instance-lookup scheme instead of an expensive spatial or instance join operation for identifying co-

location instances. But with the increasing size of co-location, the time of scanning the materialized spatial neighbour relationships will increase.

1.2.2 Attribute-Oriented Induction Methods

Attribute-Oriented Induction (AOI) (Lu et al, 1993; Ester et al, 1998; Knorr and Ng, 1997) operates by generalizing detailed spatial data to a particularly high level and studies the general characters and data distributions at this level. It has been implemented in the GeoMiner (Han et al, 1997). The goal of AOI is to discover interesting relationships between spatial and non-spatial data. There are two well known AOI algorithms: (1) AOI（*Attribute-Oriented Induction*） proposed by Cai et al (1991), and (2) *LCHR*（*Learn CHaracteristic Rule*） proposed by Han (1994). Both are not incremental and do not allow fast re-generalization. An AOI method possessing fast re-generalization was proposed by Wang (2000). But its runtime performance is not very good because it consumes too much memory space.

Carter and Hamilton (1998) proposed two new algorithms. *GDBR (Generalize Database Relation)* is an online algorithm, and *FIGR (Fast, Incremental Generalization and Re-generalization)* has characteristics of incremental and fast re-generalization. More importantly, the run times of the *GDBR* and the *FIGR* are less than the AOI and the LCHR.

But there is a supposition in the FIGR. The size of attributes and the number of the possible values in an attribute are relatively small (e.g., the size of attributes should be less than 5). In addition, the four algorithms control generalization levels by using attribute thresholds. That is not so realistic in practice, because it is impossible to try every possible combination of thresholds for every attribute. And the size of attributes and the number of the possible values in an attribute are not small in a real-world environment. So, it might not be a good idea to apply algorithms of AOI.

1.2.3 Spatial Data Fusion Methods

In the research of object fusion, Papakonstantinou, Abiteboul and Garcia-Molina (1996) and Samet, Seth and Cueto (2004) considered that objects have identifiers (e.g., keys), while Beeri et al (2004) and Minami (2000) studied this problem without global identifiers. The lack of global identifiers makes the object-fusion problem much more difficult. In addition, in the paper of Bruns and Egenhofer (1996), topological similarity is used to find corresponding objects, while Fonseca and Egenhofer (1999), Fonseca, Egenhofer and Agouris (2002), and Uitermark et al (1999) used ontology for that purpose. Finally, the problem of how to fuse objects, rather than how to find fusion sets, was studies by Papakonstantinou, Abiteboul and Garcia-Molina (1996).

Since location is the only property that is always available for spatial objects, location-based fusion problems only using object location are investigated. Minami (2000) proposed the one-sided nearest-neighbour join, Beeki et al (2004) gives the mutually-nearest method, the probabilistic method and the normalized-weights method. The mutually-nearest method is an improvement of the one-sided nearest-neighbour join, and the probabilistic method and the normalized-weights method are based on a probabilistic model which are shown in (Beeki et al, 2004) achieve the best results under all circumstances. Although these methods are very fresh and novel, they need to compute the distance between two objects. It is unfortunately not a simple task, because the locations of objects are spatial attributes.

1.3 Challenges in Spatial Data mining

Spatial data mining includes mining spatial association and co-location patterns, clustering, classification, and spatial trend and outlier analysis. The common challenges in spatial data mining are mining methodology, user interaction, performance, data type, and data size as discussed below:

1.3.1 Mining methodology and user interaction issues: These reflect the kinds of spatial knowledge miner, the ability to mine spatial knowledge at multiple granularities, the use of domain knowledge, and spatial knowledge visualization.

- *Mining different kinds of knowledge in spatial data sets*: Because different users can be interested in different kinds of spatial knowledge, spatial data mining should cover a wide spectrum of data analysis and knowledge discovery tasks, including data characterization, discrimination, association and correlation analysis, classification, prediction, clustering, outlier analysis, and evolution analysis (which includes trend and similarity analysis). These tasks may use the same spatial data set in different ways and require the development of numerous spatial data mining techniques.
- *Interactive mining of spatial knowledge at multiple levels of abstraction*: The spatial data mining process should be interactive, because it is difficult to know exactly what can be discovered within a spatial data set. For data sets containing a huge amount of spatial data, appropriate sampling techniques can be first applied to facilitate interactive data exploration. Interactive mining allows users to focus on the researching patterns, providing and refining data mining requests based on returned results. Specifically, knowledge should be mined by drilling down, rolling up, and pivoting through the data space and knowledge space interactively, similar to what OLAP can do on data cubes. In this way, the user can interact with the data

mining system to view data and discovered patterns at multiple granularities from different angles.

- *Incorporation of background knowledge*: Background knowledge, or information regarding the domain under study, may be used to guide the discovery process and allow discovered patterns to be expressed in concise terms and at different levels of abstraction. Domain knowledge related to spatial data sets, such as integrity constraints and deduction rules, can help focus and speed up a spatial data mining process, or judge the interestingness of discovered patterns.
- *Presentation and visualization of spatial data mining results:* Discovered knowledge should be expressed in high-level languages, visual representations, or other expressive forms so that the knowledge can be easily understood and directly usable by humans. This is especially crucial if the data mining system is to be interactive. This requires the system to adopt expressive knowledge representation techniques, such as trees, tables, rules, graphs, charts, crosstabs, matrices, or curves.
- *Pattern evaluation—the interestingness problem*: A spatial data mining system can uncover thousands of patterns. Many of the patterns discovered may be uninteresting to the given user, either because they represent common knowledge or lack novelty. Several challenges remain regarding the development of techniques to assess the interestingness of discovered patterns, particularly with regard to subjective measures that estimate the value of patterns with respect to a given user class, based on user beliefs or expectations. The use of interestingness measures or user-specified constraints to guide the discovery process and reduce the search space is another active area of research.

1.3.2 Performance issues: These include efficiency, scalability, and parallelization of data mining algorithms.

- *Efficiency and scalability of spatial data mining algorithms*: To effectively extract information from a huge amount of spatial data in spatial data sets, spatial data mining algorithms must be efficient and scalable. In other words, the running time of a spatial data mining algorithm must be predictable and acceptable in large spatial data sets. Considering the huge size of spatial data sets, efficiency and scalability are key issues in the implementation of spatial data mining systems. The issues discussed above under *mining methodology and user interaction* must also consider efficiency and scalability.
- *Parallel, distributed, and incremental mining algorithms*: The huge size of spatial data sets, the wide distribution of data, and the computational complexity of some spatial data mining methods are factors motivating the development of parallel and

distributed data mining algorithms. Such algorithms divide the data into partitions, which are processed in parallel. The results from the partitions are then merged. Moreover, the high cost of some data mining processes promotes the need for incremental spatial data mining algorithms that incorporate spatial data set updates without having to mine the entire data again "from scratch". Such algorithms perform knowledge modification incrementally to amend and strengthen what was previously discovered.

1.3.3 Issues relating to the spatial data type and spatial fuzzy data types

- *Handling complex spatial data types:* Spatial data mining deals with not only common data types such as integers, dates and strings, but also complex spatial data types like points, lines, and polygons. Furthermore, relationships between spatial objects, including metric (e.g., distance), directional (e.g., north of), and topological ones (e.g., adjacent), add new complexity to SDM.
- *Mining information from spatial fuzzy data sets:* If the location attribute of spatial data in a spatial data set is described as area, the spatial data set becomes a fuzzy spatial data set. Actually, the geographic proximity is a fuzzy concept in many real application fields. Discovering spatial fuzzy knowledge from spatial fuzzy data sets thus present great challenge to spatial data mining. Spatial fuzzy data mining may help disclose interesting data regularities in spatial fuzzy data sets that are unlikely to be discovered by traditional spatial data mining system.

The above issues are considered as major requirements and challenges for the investigations in spatial patterns mining. For the limited researching time, some of them will be addressed in this thesis to a certain extent, while others will be studied in the future.

1.4 Organization of the Thesis

This thesis includes the following investigations: *discovering co-location patterns from fuzzy spatial data sets*, *a new join-less approach for co-location patterns mining*, *an order-clique-based method for mining maximal prevalence co-location patterns*, *an attribute-oriented induction method based on attributes' generalization sequence*, *researching on mining prediction technologies*, *a cell-based spatial object fusion method*, *a fuzzy clustering method based on domain knowledge, and a visual spatial co-location patterns mining prototype system.* Their relationship is shown in Figure 1.1.

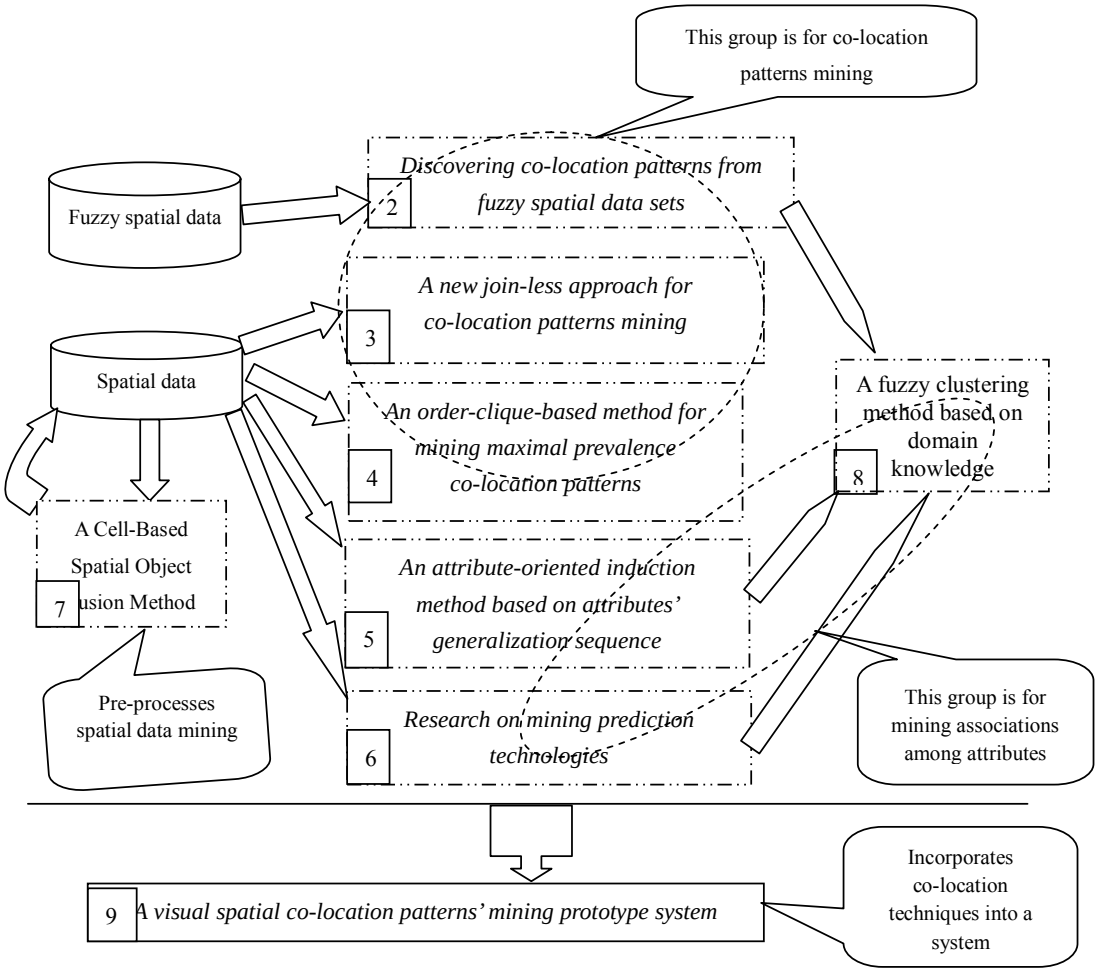


Figure 1.1 The relationship map of the contents of research in the thesis

Two types of data, fuzzy spatial data and spatial data, are the studied objects in this thesis. For fuzzy spatial data sets, the problem of discovering co-location patterns is explored (it is denoted as number 2 in Figure 1.1. It means it will be in Chapter 2.). For spatial data sets, five works are investigated. The research of the number 8 is connected to the works of the number 2, 5, and 6 since used the fuzzy equivalence partition method in the number 2 and 6 is the same as in the number 8, and applied the concept hierarchy trees in the number 5 is same as in the number 8. The efficiency of each of the techniques given in Chapter 2-8 is investigated in its own chapter. Chapter 9 gives a prototype system which incorporates the techniques for co-location patterns mining, i.e. Chapter 3 and 4. In the future, the system could be extended to incorporate techniques from other chapters, but this extension of capability is not essential to proving the efficiencies of all the techniques described in this thesis. The visual spatial co-location patterns' mining prototype system is the number 9 in Figure 1.1.

So, the rest of the thesis is composed of nine Chapters: *Discovering co-location patterns from fuzzy spatial data sets*, *a new join-less approach for co-location patterns mining, an order-clique-based method for mining maximal prevalence co-location pat-*

terns, an attribute-oriented induction method based on attributes' generalization sequence, researching on mining prediction technologies, a cell-based spatial object fusion method, a fuzzy clustering method based on domain knowledge, *a visual spatial co-location patterns' mining prototype system*, and *conclusions and concluding remarks*.

Chapter 2 provides how to d*iscover co-location patterns from fuzzy spatial data sets*. A **semantic proximity, SP**, between spatial fuzzy instances is introduced in this Chapter. Based on the fuzzy equivalence partition, the concept of co-location mining from fuzzy spatial data sets (for short, called the **fuzzy co-location mining**) is formally established. Further, an algorithm to discover the fuzzy co-location rules is designed. A new data structure, the **binary partition tree**, to improve the process of fuzzy equivalence partitioning, is proposed. A **prefix-based approach** to partition the prevalent event set search space into subsets, where each sub-problem can be solved in main-memory, is also presented. Finally, theoretical analysis and experimental results on synthetic data sets and a real-world plant distributed data set are presented and discussed.

Chapter 3 describes *a new join-less approach for identifying co-location pattern table instances.* In this Chapter, a new join-less approach for co-location patterns mining, which based on the data structure----**CPI-tree** (Co-location Pattern Instance Tree), is proposed. The CPI-tree materializes spatial neighbour relationships. All co-location instances can be generated quickly with a CPI-tree. In this chapter, the correctness and completeness of the new approach is also proved. Finally, an experimental evaluation using synthetic datasets and a real world dataset shows that the algorithm is computationally more efficient than the traditional used algorithms.

Chapter 4 discusses an order-clique-based method for mining maximal prevalence co-location patterns. In this chapter, Characteristic and efficiency of the approach is achieved with three techniques: (1) the spatial neighbour relationships between instances and the size-2 prevalence co-locations are compressed into extended prefix-tree structures respectively, ***Neib-tree*** and ***P_2-tree***, which brings up an order-clique-based approach to mining candidate maximal ordered prevalence co-locations and ordered table instances, (2) all table instances are generated from the *Neib-tree*, and **do not need be stored** after computing the *Pi* value of corresponding co-location, which dramatically reduces the executive time and space of mining maximal co-locations, and (3) some **strategies**, pruning the branches, with the number of children less than a related value, and scanning the *Neib-tree* in order, are used to avoid some useless inspection in the process of inspecting table instances.

Chapter 5 presents *an attribute-oriented induction method based on the attributes' generalization sequence*. A reasonable approach of AOI (**AOI-ags**, attribute-oriented induction based on **attributes' generalization sequence**), which expands the traditional

AOI method, is proposed in this Chapter. By introducing equivalence partition trees, an optimization algorithm of the AOI-ags is devised. Defining **interestingness of attributes' generalization sequences**, the selection problem of attributes' generalization sequences is solved. Extensive experimental results show that the AOI-ags are useful and reasonable. Particularly, by using the AOI-ags algorithm in a plant distributed dataset, some distributed rules for the species of plants in an area are found interesting.

Chapter 6 focuses on *mining prediction technologies*. Based on the concept of semantic proximity, a mining method to evaluate the **fuzzy association degree** is given in this chapter. Inverse document frequency (IDF) weight function has been adopted in this investigation to measure the weights of ecological environments in order to superpose the fuzzy association degrees. To implement the method, the "**growing window**" and the proximity computation pruning are deployed to reduce both I/O and CPU costs for the computation of the fuzzy semantic proximity between time-series. Extensive experiments on real datasets are conducted, and the results show that the mining approach is reasonable and effective.

Chapter 7 introduces *a cell-based spatial object fusion method*. This method only uses locations of objects **without calculating the distance** between two objects. The performance of the algorithm is measured in terms of recall and precision. This algorithm can work well when locations are imprecise and each spatial data set represents only some of the real-world entities. Results of extensive experimentation are presented and discussed.

A fuzzy clustering method based on domain knowledge is described in **Chapter 8**. The clustering method in this chapter is based on domain knowledge, from which the tuples' semantic proximity matrix can be obtained, and then two fuzzy equivalence partition methods are introduced. Both methods are started from semantic proximity matrix so that the results of clustering can be instructed by domain knowledge. The two methods are **Natural Method (NM)** and **Graph-Based Method (GBM)**, which are both controlled by a threshold that is confirmed by polynomial regression. Theoretical analysis testifies the correctness of the approaches. The extensive experiments on synthetic datasets compare the performance of the new approaches with that of modified MM approach in Wang (2000) and highlight the benefits of the new approaches. The experimental results on real datasets discover some rules which are useful to domain experts.

Chapter 9 focuses on the development of *a visual spatial co-location patterns' mining prototype system (**SCPMiner**).* The SCPMiner provides the user multiple methods of the spatial co-location mining. The management of co-location data is given. The co-location mining methods' analyzing can be performed on the SCPMiner. And it provides

co-location mining applying function as well. Visualization and simplicity are outstanding characteristics of SCPMiner.

Finally, In **Chapter 10**, the most important results and contributions of the thesis are concluded. In addition, some possible extensions based on current achievements are discussed. Further investigations which need to be carried out are indicated.

Chapter 2

Discovering Co-location Patterns from Fuzzy Spatial Data Sets

This Chapter extends mining spatial co-location patterns from general spatial data sets to mining spatial co-location patterns from fuzzy spatial data sets and makes the following contributions. A concept of semantic proximity SP over fuzzy spatial instances is defined. The concept of fuzzy spatial co-location mining is given based on the fuzzy equivalence partition. An algorithm for mining fuzzy spatial co-location rules is designed. A new data structure, the binary partition tree, to improve the process of fuzzy equivalence partition, is proposed. A prefix-based approach to partition the prevalent event set search space into subsets is also presented, where each sub-problem can be solved in main-memory. Finally, the time complexity and correctness of the algorithm are analyzed and experiments are conducted using synthetic data sets and a real-world plant distributed data set. A case study on real-world data sets shows that our method is effective for mining co-locations from fuzzy spatial data sets.

2.1 Overview

Spatial co-location patterns represent subsets of spatial events whose instances are often located in close geographic proximity. Spatial events describe the presence or absence of geographic object types at different locations in a two-dimensional or three-dimensional metric space, such as the surface of the earth. Examples of spatial events include plant species, animal species, business types, mobile service requests, disease, crime, climate, etc. Spatial co-location patterns may yield important insights for many applications. For example, Botanists may be interested in symbiotic plant species in a special area. E.g., “Picea Brachytyla”, “Picea Likiangensis” and “Tsuga Dumosa” grow frequently in an alpine terrain of the “Three Parallel Rivers of Yunnan Areas” zone. Co-location rules are used to infer the presence of some events (e.g., plants or animals) in the neighbourhood of instances of other spatial events. For example, “‘Picea Brachytyla’ → ‘Picea Likiangensis’ and ‘Tsuga Dumosa’ ” predicts the presence of ‘Picea Likiangensis’ and ‘Tsuga Dumosa’ plants in the areas with ‘Picea Brachytyla’.

In a plant distributed dataset, discovering spatial co-location patterns is quite a significant task. We know that plants grow in a tuft, so, the location of certain plant species is not a point but a probability area. If we approximately describe the location of plants as areas, the spatial datasets we face become fuzzy spatial datasets. How do we discover co-location patterns from a fuzzy spatial dataset while geographic proximity is not an absolute concept? For instance, if you say that certain plant is in close proximity to other

plant which is 5 meters away from it, then what is the relationship with the third plant which is 5.01 meters away? Based on the discussion above, the problems of co-location mining from fuzzy spatial datasets are investigated in this chapter (for short, called the fuzzy co-location mining).

The problem of fuzzy co-location mining can be formalized as follows: Given 1) A set E of K spatial event types $E = \{e_1, e_2, \cdots e_K\}$ and their instances $I = \{i_1, i_2, \cdots i_N\}$, each $i_i \in I$ is a vector <instance-id, spatial event type, location>, where locations are fuzzy data and they belong to a spatial framework *F*, and 2) A semantic proximity SP over instances in *I* and a fuzzy equivalence partition of instances in *I* based on SP, all the fuzzy co-location rules can be efficiently found.

2.1.1 Background of Fuzzy Co-location Mining

In previous work on mining co-location patterns, Morimoto (2001) defined distance-based patterns called k-neighbouring class sets. In his work, the number of instances for each pattern is used as the prevalence measure, which does not possess an anti-monotone property by nature. However, Morimoto used a non-overlapping instance constraint to get the anti-monotone property for this measure. In contrast, Shekhar & Huang (2001) developed an event centric model, which does away with the non-overlapping instance constraint, and a new prevalence measure called the *participation index (Pi)* is defined. This measure possesses the desirable anti-monotone property. At the same time, Huang, Shekhar & Xiong (2004) proposed a general mining approach: Join-based approach mining co-locations. This approach is good on sparse spatial data sets. However, in dealing with dense data sets, it is inefficient due to the computation time of the join is growing with the growth in co-locations and table instances. Yoo and Shekhar proposed two improved algorithms (called partial-join approach and join-less approach respectively) to conquer the disadvantage of the full-join approach on efficiency in (Yoo and Shekhar, 2004) and (Yoo et al, 2005).

Huang, Pei and Xiong address the problem of mining co-location patterns with rare spatial events in (Huang et al, 2006). In this paper, a new measure called the maximal participation ratio (maxPR) was introduced and a week monotonicity property of the maxPR measure was identified. And in paper (Huang and Zhang, 2006), Huang and Zhang proposed a new approach to the problem of mining co-location patterns using clustering techniques. Therefore, clustering techniques can be applied to reveal the rich structure formed by co-located spatial events in spatial data.

In summary, the problem of mining spatial co-location patterns is being widely investigated from the measures, algorithms to application domains, but not in fuzzy co-location mining.

2.1.2 Organization of the Chapter

The remainder of the Chapter is organized as follows: Section 2.2 presents basic concepts of the fuzzy co-location mining. In Section 2.3, an algorithm for fuzzy co-location mining is presented. Section 2.4 provides an analysis of the algorithms in the area of correctness, completeness and computational complexity. Experimental evaluations are given in Section 2.5. The conclusion and discussing future work are given in Section 2.6.

2.2 Definitions of Basic Concepts

This section defines the basic concepts of the fuzzy co-location mining. Figure 2.1 is used as an example to illustrate these concepts. In Figure 2.1, each instance is uniquely identified by $E.i$, where E is the spatial event type, and i is the unique id inside each spatial event type, i.e., $A.2$ represents the second instance of spatial event type A.

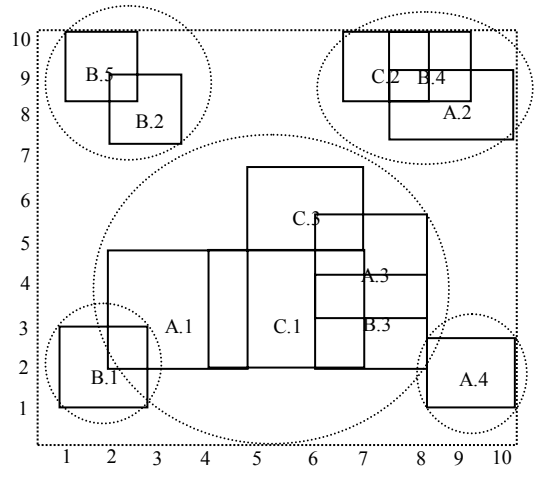


Figure 2.1 An example of spatial event instances

Table 2.1 An example of spatial instances (Instances set sorted by spatial event types and instance-id)

Instance-id	Spatial event type	Location
1	A	([2-5], [2-5])
2	A	([7-10], [7-9])
3	A	([6-8], [3-6])
4	A	([8-10],[1-3])
1	B	([1-3], [1-3])
2	B	([2-4],[7-9])
3	B	([6-8], [2-4])
4	B	([7-9] ,[8-10])
5	B	([1-3], [8-10])
1	C	([4-7], [2-5])
2	C	([6-8],[8-10])
3	C	([5-7], [5-7])

Instances can be described as a vector <instance-id, spatial event type, location> (see the Table 2.1). The location of an instance is a fuzzy value and belongs to a spatial framework *F*.

Zadeh provides the requisite mathematical framework for handling fuzzy data values in (Zadeh, 1965). A fuzzy subset $\widetilde{X}_i$ in *F* is characterized by a membership function

$\mu_{\widetilde{X}_i} : F \to [0,1]$. $\mu_{\widetilde{X}_i}(x)$, for each $x \in F$, denotes the grade of membership of x in the fuzzy subset $\widetilde{X}_i$.

The interval number description and centre number description are the two most common examples of Zadeh's descriptions (Liu, 1993).

The interval number description:

A fuzzy subset $\widetilde{X}_i$ is characterized by an ordered couple $[a-b]/\delta$. $[a-b]$ is called an interval number. It expresses the fact that this fuzzy subset lies between a and b. $\delta(0 \le \delta \le 1)$ is the degree of confidence. The subsets are stacked according to the confidence degrees.

The centre number description:

A fuzzy subset $\widetilde{X}_i$ is characterized by $[c,r]/\delta$. It expresses that this fuzzy subset lies in a spherical region. c is the centre of a sphere, r is its radius, δ as above. For instance, we say the length of a string is 10.17±0.03cm.

The interval number description is selected in this chapter. However, the same method can be used to deal with the centre number description and the Zadeh descriptions.

For the sake of convenience, the confidence degree of fuzzy values is sometimes omitted. It means that the confidence degree of each fuzzy value is united into 1. For example, suppose the probability distribution of values is a normal distribution. Since $\int_{-3\sigma}^{3\sigma} f(x)dx$=0.99, $[a-b]/\delta$ is simplified by $[-3\sigma-3\sigma]$, where σ is the standard deviation (Amstader, 1979). Suppose the probability distribution of values is evenly distributed. $[a-b]/\delta$ is simplified by $[(a-\beta/2)-(b+\beta/2)]$, where $\delta/(b-a)=1/\beta$. In case the probability distribution of values is unknown, for convenience, it is regarded as evenly distributed. The confidence degree of a classical value is 1. For example, 2.7 is denoted $[2.7-2.7]$. This is a simple and intuitive method. The exact method is not discussed in this chapter.

How to define the proximity between spatial instances which's locations are presented by a fuzzy value as shown in table 2.1 is an important issue in fuzzy co-location mining. Based on the concept of the interval number of fuzzy values, the semantic proximity SP is introduced to define the geographic proximity between instances.

2.2.1 The semantic proximity *SP* and the fuzzy equivalence partition

The **semantic proximity** is the degree of proximity between instances f_1 and f_2 (their locations are $f_1 = ([a_1^1 - a_1^2],[b_1^1 - b_1^2])$ and $f_2 = ([a_2^1 - a_2^2],[b_2^1 - b_2^2])$, written $SP(f_1, f_2)$ $(0 \le SP(f_1, f_2) \le 1)$, can be defined as

$$SP(f_1, f_2) = (Area(f_1 \cap f_2) / Area(f_1 \cup f_2)) \qquad (1)$$

where $Area(h)$ is the area of rectangle h.

Example 2.1. Suppose $f_1 = ([2\text{-}5],[2\text{-}5])$, $f_2 = ([1\text{-}3],[1\text{-}3])$. Then

$SP(f_1, f_2) = (1/12) = 0.083$

The semantic proximity (*SP*) between two fuzzy values defined by formula (1) satisfies the following properties.

(1). If f_1, f_2 are two equal fuzzy values then the SP of f_1 and f_2 is 1.

(2). If f_1, f_2 are two locations that do not intersect, then the SP of f_1 and f_2 is 0.

(3). If the area of the location f_1 is equal to the area of the location f_2, and the area of $f_1 \cap g_1$ is greater than the area of $f_2 \cap g_1$ then $SP(f_1,g_1)$ is greater than $SP(f_2,g_1)$.

(4). If the area of the $f_1 \cap g_1$ is equal to the area of the $f_2 \cap g_2$, and the area of $f_1 \cup g_1$ is greater than the area of $f_2 \cup g_2$ then $SP(f_1,g_1)$ is smaller than $SP(f_2,g_2)$.

There are many expressions of the proximity from various angles (He, 1989; Liu, 1993; Liu and Song, 2001; Schwartz, 1989; Ziarko, 1991). Which one you choose depends on your applications. The different expression would not affect the following discussion.

Considering the proximity does not satisfy transitivity, **the fuzzy equivalence partition** method is introduced in geographic proximity instances. Assume that $i_1, \cdots, i_N$ is a sequence of instances of events. From the above points, $SP(i_i, i_i) = 1$ and $SP(i_i, i_j) = SP(i_j, i_i)$ hold. Using the semantic proximity between spatial instances, a similarity matrix $S = (s_{ij})_{N \times N}$ can be built up in (2):

$$S = \begin{cases} 1 & i = j \\ SP(i_i, i_j) & i \ne j \end{cases} \qquad (2)$$

The matrix *S* is multiplied by itself repeatedly, where $(s_{ij})^2 = MAX_k(MIN(s_{ik}, s_{kj}))$, until $S^{2k} = S^k$. S^{2k} is called a fuzzy equivalence matrix (i.e., $s_{ij}^k = 1, s_{ij}^k = s_{ji}^k, S^{2k} = S^k$) (Wang, 2000; Huo, 1989).

Based on the level value matrix of the fuzzy equivalence matrix, the classifications of $i_1, \cdots, i_N$ can be obtained (Wang, 2000; Huo, 1989).

Example 2.2. Based on the definition of *SP* of two spatial instances, the similarity matrix *S* can be obtained as (3) from Figure 2.1.

$$S = \begin{bmatrix} 1 & 0 & 0 & 0 & 0.083 & 0 & 0 & 0 & 0 & 0.2 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0.25 & 0 & 0 & 0.111 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0.25 & 0 & 0 & 0.154 & 0 & 0.111 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.083 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0.143 & 0 & 0 & 0 \\ 0 & 0 & 0.2 & 0 & 0 & 0 & 1 & 0 & 0 & 0.182 & 0 & 0 \\ 0 & 0.25 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0.333 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.143 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0.2 & 0 & 0.154 & 0 & 0 & 0 & 0.182 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0.111 & 0 & 0 & 0 & 0 & 0 & 0.333 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0.111 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (3)$$

Applying S self-multiply repeatedly, the obtained matrix is shown in (4):

$$EM = S^8 = S^4 = \begin{bmatrix} 1 & 0 & 0.182 & 0 & 0.083 & 0 & 0.182 & 0 & 0 & 0.2 & 0 & 0.111 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0.25 & 0 & 0 & 0.25 & 0 \\ 0.182 & 0 & 1 & 0 & 0.083 & 0 & 0.25 & 0 & 0 & 0.154 & 0 & 0.111 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.083 & 0 & 0.083 & 0 & 1 & 0 & 0.083 & 0 & 0 & 0.083 & 0 & 0.083 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0.143 & 0 & 0 & 0 \\ 0.182 & 0 & 0.2 & 0 & 0.083 & 0 & 1 & 0 & 0 & 0.182 & 0 & 0.111 \\ 0 & 0.25 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0.333 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.143 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0.2 & 0 & 0.182 & 0 & 0.083 & 0 & 0.182 & 0 & 0 & 1 & 0 & 0.111 \\ 0 & 0.25 & 0 & 0 & 0 & 0 & 0 & 0.333 & 0 & 0 & 1 & 0 \\ 0.111 & 0 & 0.111 & 0 & 0.083 & 0 & 0.111 & 0 & 0 & 0.111 & 0 & 1 \end{bmatrix} \quad (4)$$

Then, S^4 is the fuzzy equivalence matrix *EM* of the similarity matrix *S*. If selecting level value λ=0.09, the level value matrix $S^4_{0.09}$ is obtained as (5) (the value becomes 1 if it is greater than λ, otherwise zero).

$$S^4_{0.09} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \end{bmatrix} \quad (5)$$

A partition of spatial instances in Figure 2.1 can be obtained from $S^4_{0.09}$: $I_{0.09}$={*(A.1, A.3, B.3, C.1, C.3), (A.2, B.4, C.2), (A.4), (B.1), (B.2, B.5)}.* This result is described by the dashed circles in Figure 2.1.

2.2.2 Further Definitions Based on the *SP* and fuzzy equivalence partition

Based on the concepts of the fuzzy semantic proximity *SP* and fuzzy equivalence partition, other concepts can be defined as follows.

Given *I* is an instance set of event set *E*, a **semantic proximity neighbourhood** is a set $I' \subset I$ of instances that belong to a fuzzy equivalence class.

A **co-location *C*** is a subset of spatial events, i.e., $C \subseteq E$. A **co-location rule** is of the form: $C_1 \to C_2(p, cp)$, where C_1 and C_2 are disjoint co-locations, p is a value representing the prevalence measure, and cp is the conditional probability.

A semantic proximity neighbourhood I' is a **row instance** (denoted by row-instance (C)) of a co-location C if I' contains instances of all the events in C and no proper subset of I' does so. The **table instance**, table-instance (C), of a co-location C is the collection of all row instances of C.

Example 2.3. Suppose the dashed circles in Figure 2.1 represent fuzzy equivalence classes. In Figure 2.1, we observe that $\{A.3, B.3\}$ is a row instance of co-location $\{A, B\}$. $\{A.3, C.1, C.3\}$ is a semantic proximity neighbourhood, but it is not a row instance of co-location $\{A, C\}$ because its subset $\{A.3, C.1\}$ or $\{A.3, C.3\}$ contain instances of all the events in $\{A, C\}$. The table instance of $\{B, C\}$ has 3 row instances $\{B.3, C.1\}$, $\{B.3, C.3\}$ and $\{B.4, C.2\}$.

From the definitions above, it can be observed that the concept of semantic proximity neighbourhood is not an absolute concept. The geographic proximity relationship can be controlled by changing the fuzzy equivalence partition threshold λ (the level value λ). Further more, because of the fuzzy equivalence partition, the problem of high cost that is happened during computing the table instance in traditional co-location mining can be improved in fuzzy co-location mining. However, the results of a fuzzy equivalence classification are not equal to that of transactions, because there are many row instances of a co-location in a fuzzy equivalence class. So, the following definitions are similar to the definitions given by Huang et al (2004).

The ***conditional probability*** $CP(C_1 \Rightarrow C_2)$ of a co-location rule $C_1 \Rightarrow C_2$ is the probability of finding an instance of C_2 in the semantic proximity neighbourhood of an in-

stance of C_1. Formally, it is estimated as $\frac{|\pi_{C_1}(table-\text{instance}(\{C_1 \cup C_2\}))|}{|table-\text{instance}(\{C_1\})|}$, where π is the relational projection operation with duplicate elimination.

The ***participation index*** is used as a co-location prevalence measure. The participation index $Pi(C)$ of a co-location $C=\{e_1,\cdots,e_k\}$ is defined as $\min_{e_i \in C}\{\Pr(C,e_i)\}$, where $\Pr(C,e_i)$ is the ***participation ratio*** for event type e_i in a co-location C. $\Pr(C,e_i)$ is the fraction of instances of e_i which participate in any instance of co-location C, $\frac{|\pi_{e_i}(table-\text{instance}(\{C\}))|}{|table-\text{instance}(\{e_i\})|}$, where π is the relational projection operation with duplication elimination.

Example 2.4. In Figure 2.1, the total number of instances of event type B is 5 and the total number of instances of event type C is 3. The participation index of co-location $c=[B,C]$ is $\min\{\Pr(c,B),\Pr(c,C)\}=2/5$, because $\Pr(c,B)$ is 2/5 and $\Pr(c,C)$ is 3/3.

In Huang's work (Huang et al, 2004), it can be known that the participation ratio and the participation index are monotonically non-increasing with the increase in size of the co-location. So, the participation index can be used to effectively prune the search space of co-location rules mining.

2.3 Algorithms for Discovering Fuzzy Co-location

In this section, an algorithm to mine fuzzy co-location rules is introduced. The inputs of this algorithm are a set *E* of spatial event types, a set *I* of spatial instances, a user-specified level value λ as well as thresholds for interest measures, i.e., minimum prevalence threshold, *min_prev,* and conditional probability threshold, *min_cond_prob*. The algorithm outputs a set of prevalent fuzzy spatial co-location rules with the values of the interest measures above the user-defined thresholds. The detailed descriptions are shown as follows.

Input

E: a set of K spatial event types;

I: a set of N instances <event type, event instance id, and fuzzy location>;

λ : a user-specified level value for controlling the fuzzy equivalence partition;

min_prev: prevalence value threshold;

min_cond_prob: conditional probability threshold;

Output

A set of all prevalence co-location rules with participation index greater than *min_prev* and conditional probability greater than *min_cond_prob*.

Variables

k: co-location size;

C_k: set of candidate co-locations of size k;

T_k: set of table instances of co-locations in C_k;

P_k: set of prevalent co-locations of size k;

R_k: set of co-location rules of size k;

S: matrix of semantic proximity between instances;

EM: fuzzy equivalence matrix for the fuzzy similarity matrix S;

EPC: fuzzy equivalence classifications for a set I of N instances;

Steps

1) Takes E, I, λ, *min_prev* and *min_cond_prob*;
2) Computing the semantic proximity between instances, a similarity matrix $S = (s_{ij})_{N\times N}$ can be obtained, where $s_{ij} = SP(e_{i.location}, e_{j.location})$, $s_{ii} = 1$, $s_{ij} = s_{ji}$, $i, j = 1,2,\cdots N$;
3) Calculate a fuzzy equivalence matrix EM from the similarity matrix S;
4) Based on user-specified level value λ, the classifications $EPC = \{s_1, s_2, \cdots, s_l\}$ for a set I of N instances can be obtained;
5) k:=1; C_1:=E; P_1:=E;
6) T_1=gen_table_instance (C_1, I, EPC);
7) **While** (not empty P_k and k<K) **do {**
8) C_{k+1}=gen_candidate_co-location (P_k);
9) T_{k+1}=gen_table_instance (C_{k+1}, T_k);
10) P_{k+1}=select_prevalence_co-location (*min_prev*, C_{k+1}, T_{k+1});
11) R_{k+1}=gen_co-location_rule (*min_cond_prob*, P_{k+1});
12) k:=k+1; **}**
13) **Return** $\cup(R_2, \cdots, R_{K+1})$;

Step 1, i.e., input step, takes E, I, λ, min_prev, and min_cond_prob. Step 2, compute semantic proximity between instances. A similarity matrix S is obtained. Step 3, self-multiply the similarity matrix S repeatedly, and then obtain the fuzzy equivalence matrix EM. The equivalence matrix EM can be computed in $O(N^3)$ time. The computational method can be expressed as:

```
1) For i:=1 to N do
2)   For j:=1 to N do
3)     If s(j,i)>0 then
4)       For k:=1 to N do
5)         S(j,k):=max{S(j,k), min{S(j,i), S(i,k)}};
```

where the test s(j,i)>0 in Step 3 is to avoid meaningless looping. This algorithm has high efficiency when the fuzzy similarity matrix *S* has many zero elements.

The above method can be optimized. Wang (2000) proposed an algorithm to compute the fuzzy equivalence matrix *EM* in $O(M^2)$ time (M is the number of nonzero elements in the upper-triangle of the similarity matrix *S*.).

Suppose the spatial event instances can correspond to integer 1~N (the N entries instances can be stored in an array, and then the N instances will correspond to the index of the array). So, the N instances can be denoted as $1,2,\cdots,N$.

A new data structure, ***the binary partition tree***, to store fuzzy equivalence matrix *EM,* is introduced. A binary partition tree for N instances will have N leaf nodes and at most N-1 inner nodes. The relations between nodes are joined by parents' relations. The leaf nodes are denoted as $1,2,\cdots,N$, and inner nodes are denoted as $N+1, N+2,\cdots,2N-1$. A function *F[i]* is defined, which values express transitive semantic proximity between corresponding leaf nodes.

A link is used to store the binary partition tree, and put it to an array T. Node *i* is put into *T[i]*. So, every node has only one field, which is the parent field.

In this algorithm, 3 arrays *v [N+1,N+M], w [N+1,N+M], a [N+1,N+M]* are used to store all nonzero elements in the upper-triangle of the similarity matrix *S*, where '*v*' contains row coordinates, '*w*' contains column coordinates, '*a*' contains corresponding values, and M is the number of nonzero elements. Therefore $((v_j, w_j), a_j) \in (\{1,..., N\} * \{1,..., N\}) * [0,1]$. The new algorithm for computing the equivalence partition matrix can be designed as:

```
1)  n1:=N+1;  nm:=N+M;
2)  for i:=1 to 2N-1 do T(i):=0;
3)  sort((vj,wj),aj)    /*j=n1,…,nm; take nonzero elements in upper-triangle by aj≥aj+1, j=n1,…, nm-1*/
4)  k:=n1;
5)  for i:=n1 to nm do
        {
6)      Pv:=v(i);
7)      While T(Pv)>0 do Pv:=T(Pv);    /*tracing its parents*/
8)      Pw:=w(i);
9)      While T(Pw)>0 do Pw:=T(Pw);
10)     if Pv≠Pw then
            {
11)         T(Pv):=T(Pw):=k;  /*if they have no common parent, then produce a parent and
give it a transitive semantic proximity*/
12)         F(k):=a(i);
13)         k:=k+1
            }
        }
```

Example 2.5. An example to illustrate the algorithm above

Suppose a similarity matrix S is as (6).

$$S = \begin{bmatrix} 1 & 0.71 & 0 & 0 & 0.36 & 0 & 0 & 0 & 0 & 0 \\ 0.71 & 1 & 0 & 0.36 & 0 & 0.36 & 0.36 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0.71 & 0 & 0 & 0.36 & 0.36 & 0.79 \\ 0 & 0.36 & 0 & 1 & 0 & 0.42 & 0.59 & 0 & 0.24 & 0 \\ 0.36 & 0 & 0.71 & 0 & 1 & 0 & 0 & 0 & 0 & 0.71 \\ 0 & 0.36 & 0 & 0.42 & 0 & 1 & 0.42 & 0 & 0.71 & 0 \\ 0 & 0.36 & 0 & 0.59 & 0 & 0.42 & 1 & 0 & 0.24 & 0 \\ 0 & 0 & 0.36 & 0 & 0 & 0 & 0 & 1 & 0.36 & 0.36 \\ 0 & 0 & 0.36 & 0.24 & 0 & 0.71 & 0.24 & 0.36 & 1 & 0.36 \\ 0 & 0 & 0.79 & 0 & 0.71 & 0 & 0 & 0.36 & 0.36 & 1 \end{bmatrix} \quad (6)$$

After the algorithm carrying out step 1), 2) and 3), the results of the arrays *v, w, a* and the link *T* are shown in Figure 2. 2. Figure 2.3 demonstrates that the changing situations of *Pv* and *Pw*, and the content of the link *T*, when the loop iterating step 5) is performed. Figure 2.4 is the binary partition tree in the end (it is stored by a link *T*).

										n	n1												
i:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	…
T=	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	…
										v=	3	1	3	5	6	4	4	6	1	2	2	2	…
										w=	10	2	5	10	9	7	6	7	5	4	6	7	…
										a=	0.79	0.71	0.71	0.71	0.71	0.59	0.42	0.42	0.36	0.36	0.36	0.36	…

Figure 2.2 The result after implementing the step 1), 2) and 3)

i	*k*	*Pv*	*Pw*
11	11	3	10
12	12	1	2
13	13	3	5
		11	
14	14	5	10
		13	13
15	14	6	9
16	15	4	7
17	*16*	4	6
		15	14
…		…	…

i:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
T=	12	12	11	15	13	14	15		14	11	13			16	16	

F(11)=0.79 *F(12)=0.71* *F(13)=0.71* *F(14)=0.71* *F(15)=0.59*

F(16)=0.42

Figure 2.3 The partial results in executing the loop iterating step 5)

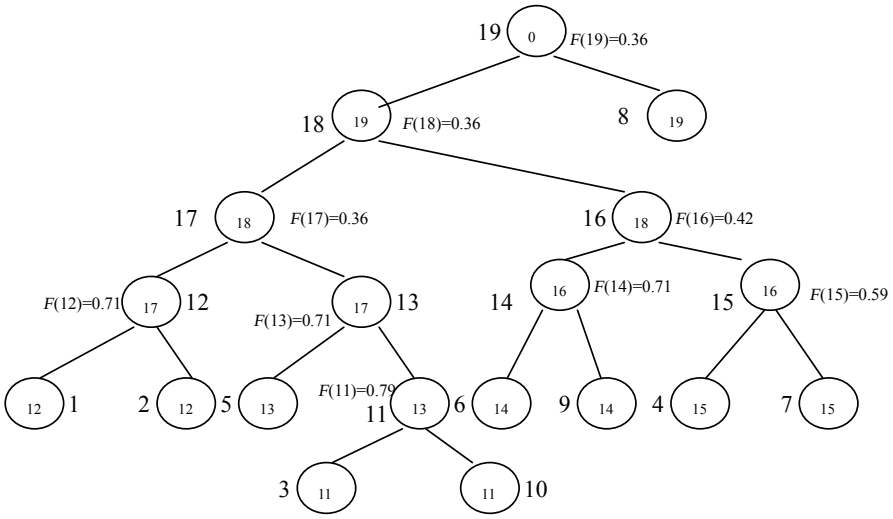


Figure 2.4 The binary partition tree corresponding to the matrix S in Example 2.5

Based on the fuzzy equivalence matrix *EM* or the binary partition tree, the equivalence partition classifications *EPC* for user-specified level value λ will be obtained in step 4.

Step 5 is the initialization step that assigned starting values to various data structures used in the algorithm. It can be noted that the set C_1 of candidate co-location of size 1 as well as the set P_1 of prevalent co-locations are initialized to E, the set of spatial event types. Because the value of the participation index is 1 for all co-location of size 1, the set T_1 of table instances of size 1 co-location is created by sorting the class-id in EPC and the set I of spatial instances by event types (see Figure 2.5 (a)).

The proposed algorithm of fuzzy co-location mining iteratively performs four basic tasks, namely, generation of candidate co-locations, generation of table instances of candidate co-locations, selection of prevalent co-locations, and generation of co-location rules. These tasks are carried out inside a loop iterating over the size of the co-locations.

2.3.1 Generation of Candidate Co-locations

The *apriori_gen* (Agarwal and Srikant, 1994) is used for generating candidate co-locations. Size *k+1* candidate co-locations are generated based on size *k* prevalence co-locations. The anti-monotonic property of the participation index makes event level pruning feasible (Huang et al, 2004) (see Figure 2.5 (b) (d)).

2.3.2 Generation of Table Instances

The table instances of candidate co-locations are enumerated using the method that is similar to the *apriori* algorithm. It can be described as the following join query.

Forall co-location $c \in C_{k+1}$

Insert into T_c /* T_c is the table instance of co-location c */

Select $p.\text{instance}_1, \cdots, p.\text{instance}_k, q.\text{instance}_k$

From c.table_instance_id$_1$ p, c.table_instance_id$_2$ q

Where p.instance$_1$=q.instance$_1$,…, p.instance$_{k-1}$=q.instance$_{k-1}$,

$p.ins\tan ce_k, q.ins\tan ce_k \in s_i$, i=1, …, *l* /*$s_i$ is a fuzzy equivalence class*/

End

The combinatorial join predicate (i.e., $p.\text{instance}_1 = q.\text{instance}_1, \cdots p.\text{instance}_{k-1} = q.\text{instance}_{k-1}$) can be processed efficiently using a sort-merge join strategy proposed by Graefe (1994), since the set of events is ordered and tables' c.table_instance_id$_1$ and c.table_instance_id$_2$ are sorted. The resulting tuples are checked for the spatial condition $P.\text{instance}_k, q.\text{instance}_k \in s_i, i = 1, \cdots l$ to get the row-instance in the result.

For accelerating the joining operations of enumerating table instances, a **class-id** is added to the table instance, by which and the instance-id the set of table instances is ordered, then the instances that belong to a fuzzy equivalence class will be joined efficiently.

Example 2.6. In Figure 2.5, row instance {1,3} of Tab. 4 and row instance {1,1} of Tab. 5 are joined to generate row instance {1,3,1} of co-location {A, B, C} (Tab. 7). Row instance {1} of Tab. 1 and row instance {1} of Tab. 2 fail to generate row instance {1,1} of co-location {A, B} because instance 1 of A and instance 1 of B do not belong to a fuzzy equivalence class.

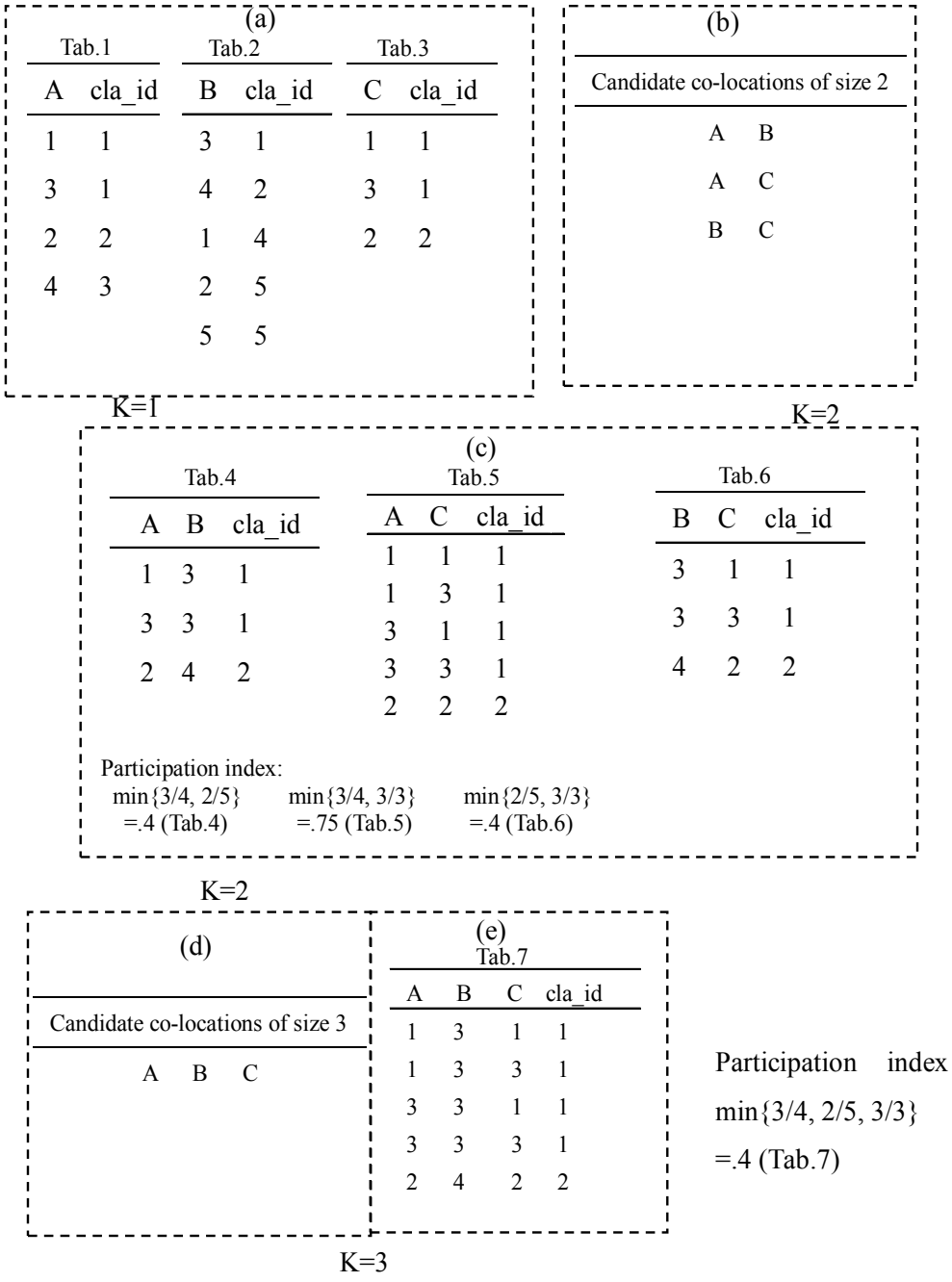


Figure 2.5 An illustration example of the fuzzy co-location mining algorithm

If the number of events is enormous, the limit of main-memory could be a problem. As the number of the species of plants located in “Three Parallel Rivers of Yunnan Protected Areas” is abound, this problem must be resolved. This, thus, brings up the issue: Can the original event set be decomposed into smaller pieces so that each partition can be solved independently in main-memory? The related discussion is addressed below:

Let E be a set of events. The power set $\mathrm{P}(E)$ of E is a complete lattice, which is proved by Zaki (2000). Figure 2.6 shows the power set lattice $\mathrm{P}(E)$ of the events' set E={A, B, C, D}. Define a function

$$f : \mathrm{P}(E) \times N \to \mathrm{P}(E) \qquad (7)$$

where $f(X,k) = X[1:k]$, the k lengths prefix of X. Define a relation θ_k on the lattice $\mathrm{P}(E)$ as follows:

$$\forall X, Y \in \mathrm{P}(E), X \equiv_{\theta_k} Y \Leftrightarrow f(X,k) = f(Y,k) \qquad (8)$$

That is, two event sets are in the same class if they share a common k length prefix. Therefore θ_k is called as a prefix-based relation. In fact, θ_k is an equivalence relation, because it is a reflexible, symmetric and transitive relation. The equivalence relation partitions a set into disjointed subsets called equivalence classes. The equivalence class of an element $X \in P(E)$ is given as $[X] = \{Y \in P(E) \mid X =_{\theta_k} Y\}$.

Figure 2.6 shows the equivalence classes induced by the equivalence relation θ_1 on $\mathrm{P}(E)$, where all power sets of events are collapsed with a common, length one, prefix into an equivalence class. The resulting set of equivalence classes is {[A], [B], [C], [D]}.

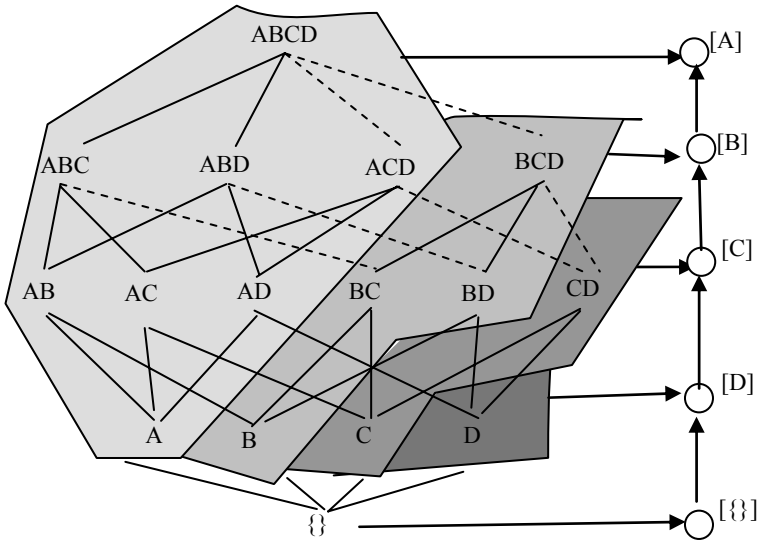


Figure 2.6 The power set lattice $\mathrm{P}(E)$ of the events' set *E={A, B, C, D}*, and the lattice induced by equivalence relation θ_1 on $\mathrm{P}(E)$.

Lemma 2.1 *Each equivalence class* $[X]_{\theta_k}$ *induced by the equivalence relation* θ_k *is a sub-lattice of* $\mathrm{P}(E)$.

Proof. (of Lemma 2.1). Let U and V be any two elements in the class [X]. i.e., U, V share the common prefix X. $U \vee V = U \cup V \supseteq X$ implies that $U \vee V \in [X]$, and $U \wedge V = U \cap V \supseteq X$ implies that $U \wedge V \in [X]$. Therefore, $[X]_{\theta_k}$ is a sub-lattice of $\mathrm{P}(E)$. □

Each $[X]_{\theta_i}$ is itself a Boolean lattice with its own set of atoms. For example, the atoms of $[A]_{\theta_1}$ are {AB, AC, AD}, and the top and bottom elements are $\mathrm{T} = ABCD$, and $\perp = A$. If there is enough main-memory to hold instance sets for each class, then each $[X]_{\theta_i}$ can be solved independently. Another interesting feature of the equivalence classes is that the links (including real links and dashed links) between classes denotes dependencies. That means if you want to put pruning in to practice, then the classes have to be processed in a specific order. In particular, the classes have to be resolved from bottom to top, which corresponds to a reverse lexicographic order, for example, in Figure 2.6, we process them in the order of [D], [C], [B], and [A]. This guarantees that all subset information is available for pruning.

In addition, depending on the amount of main-memory available, one can recursively partition large classes into smaller ones (by using θ_2, θ_3, …) until each class is small enough to be solved independently in main-memory. As it happens, according to the sort-merge join method developed in this chapter, the joining operations for enumerating table instances will be in a partition class. The real links denote join operation for enumerating table instances in Figure 2.6.

2.3.3 Selection of Prevalent Co-locations

The participation index of co-locations is calculated by scanning the table instances once. First, keep a bitmap of size cardinality *(e_i)* for each event e_i of co-location *c*. Second, scan of the table instance of *c*, and put the table instance of *c* in the corresponding bits in each bitmap. Finally, obtain the participation ratio of each event e_i (divide p_{e_i} by instance of e_i) by summarizing the total number of ones (p_{e_i}) in each bitmap.

Example 2.7. Computing the participation index of co-location {A, C}

First, bitmap b_A=(0,0,0,0) of size 4 for A and bitmap b_C=(0,0,0) of size 3 for C are initialized to zeros (see Figure 2.5 (c) Tab. 5).

Second, scan Tab. 5 and get the following results: b_A=(1,1,1,0) and b_C=(1,1,1).

Finally, three out of four instances of A participate in co-location {A, C}, so the participation ratio for A is 0.75. Similarly, the participation ratio for C is 1.00. Therefore, the participation index is min {0.75, 1.00}=0.75.

After the participation index for each co-location is determined, the selection of prevalent co-location is carried out. In other words, the nonprevalent co-locations are deleted from the candidate prevalence co-location sets. For example, if *min_prev* is given as 1/2, the candidate co-location {A, B} and {B, C} is pruned in the first loop iterating because its prevalence measure is less than 1/2.

2.3.4 Generating Co-location Rules

The *gen_co-location_rule* function generates all co-location rules with conditional probability above a given *min_cond_prob*. The conditional probability of a co-location rule can be calculated efficiently by using bitmaps strategies that have been used in Section 2.3.3.

2.4 Analysis for Fuzzy Co-location Mining

In this section, the completeness, correctness and computational complexity of the algorithm are analyzed. Correctness means that the participation index values and conditional probability of generated co-location rules meet the user specified threshold. Completeness implies that no co-location rule, which satisfy given prevalence and conditional probability thresholds, is missed.

2.4.1 Completeness and Correctness

Lemma 2.2. *The participation ratio and the participation index are monotonically non-increasing with increasing the size of the co-location.*

Proof. (of lemma 2.2). According to the definition of semantic proximity neighbourhoods, row instances and table instances in Section 2, it can be known that a spatial event instance that participates in a row instance of a co-location C also participates in a row instance of a co-location C', where $C' \subseteq C$ because a subset of an instance set that belongs to a fuzzy equivalence class must belong to this equivalence class. So, the participation ratio is anti-monotonic. The proof of anti-monotone of the participation index is shown as follows.

Suppose a co-location $C = \{e_1, \cdots, e_k\}$, then

$$Pi(C \cup e_{k+1}) = \min_{i=1}^{k+1} \{\Pr(C \cup e_{k+1}, e_i)\} \le \min_{i=1}^{k} \{\Pr(C \cup e_{k+1}, e_i)\}$$

$$\le \min_{i=1}^{k} \{\Pr(C, e_i)\} = Pi(C) \qquad \square$$

Lemma 2.3. *The Fuzzy Co-location Miner algorithm is correct.*

***Proof*.** (of lemma 2.3). First, the value of the participation index is 1 for all fuzzy co-locations of size 1, so it is correct that the set C_1 of candidate co-locations of size 1 as well as the set P_1 of prevalent co-location are initialized to E in step 5.

Then, for the size greater than 2, it will only be shown that the row instances of each co-location are correct, which will imply the correctness of the participation index values and that each co-location meets the user specified threshold. An instance $I_1 = \{i_{1,1}, \cdots i_{1,k}\}$ of $c_1 = \{e_1, \cdots, e_k\}$ and an instance $I_2 = \{i_{2,1}, \cdots i_{2,k}\}$ of $c_2 = \{e_1, \cdots, e_{k-1}, e_{k+1}\}$ are joined to produce an instance $I_{new} = \{i_{1,1}, \cdots i_{1,k}, i_{2,k}\}$ of $c = \{e_1, \cdots, e_{k+1}\}$ if 1) all elements of I_1 and I_2 are the same except $i_{1,k}$ and $i_{2,k}$; 2) $i_{1,k}$ and $i_{2,k}$ are in the same fuzzy equivalence class. The schema of I_{new} is apparently c, and elements in I_{new} are in a fuzzy equivalence class because the intersection set between fuzzy equivalence classifications is ϕ, $i_{2,k}$ and $i_{1,1}, \cdots i_{1,k-1}$ belongs to a fuzzy equivalence class, at the same time, also $i_{2,k}$ and $i_{1,k}$ belong to a fuzzy equivalence class. □

Please refer to Wang's paper (Wang, 2000) for the correctness of the fuzzy equivalence partition algorithm using binary partition tree.

Lemma 2.4. *The Fuzzy Co-location Miner is complete.*

Proof. (of lemma 2.4). It will be proved that if a fuzzy co-location is prevalent, it is found by the algorithm. First, the monotonicity of the participation index in Lemma 2.2 ensures the completeness of the event level pruning of candidate co-location used in step 8 of the algorithm. Second, it will be proved that the joining of the table instances of C_1 and C_2 to produce the table instances of C is complete in step 9. According to the semantic proximity neighbourhood definition, any subset of a semantic proximity neighbourhood is a semantic proximity neighbourhood too. For any instance $I = \{i_1, \cdots i_{k+1}\}$ of co-location C, subsets $I_1 = \{i_1, \cdots i_k\}$ and $I_2 = \{i_1, \cdots i_{k-1}, i_{k+1}\}$ are neighbourhoods, i_k and i_{k+1} are in the same fuzzy equivalence class, and I_1 and I_2 are row instances of C_1 and C_2, respectively. Join I_1 and I_2 will produce I. In step 10, enumeration of the subsets of each of the prevalent co-locations ensures that no spatial fuzzy co-location rules with both higher the *min_prev* and the *min_cond_prob* are missed. □

2.4.2 Computational Complexity Analysis

The computational complexity of the fuzzy co-location mining algorithm consists of two parts. One is outside iteration, and the other is inside iteration.

For outer part of the iterating loops, steps 1, 5 and 6 are initialized. Their computational complexity is O(K+N). The steps 2, 3, calculate the similarity matrix *S*, equivalence matrix *EM* and obtain fuzzy equivalence classification *EPC*. These can be computed in $O(M^2)$ time, if the method of binary partition tree is used. So, the computational complexity of this part is $O(M^2)$.

For inner part of the iterating loops, let $T_{fcm}(k+1)$ represent the cost of iteration *k* of the co-location miner algorithm.

$$T_{fcm}(k+1) = T_{gen\_candi}(P_k) + T_{gen\_inst}(table\_insts \quad of \quad P_k) + T_{prune}(C_{k+1})$$
$$\approx T_{gen\_inst}(table\_insts \quad of \quad P_k)$$

In above equations, $T_{gen\_candi}(P_k)$ represents the cost of generating size k+1 candidate co-locations with the prevalent size k co-locations. $T_{gen\_inst}(table\_insts \quad of \quad P_k)$ represents the cost of generating table instances of size k+1 candidate co-locations with size k table instances. $T_{prune}(C_{k+1})$ represents the cost of pruning non prevalent size k+1 co-location.

The bulk of the time is consumed in generating table instances. For generating table instances of size-2 co-locations, the complexity is $O(N^2)$. It is difficult to express the time which takes to generate table instances of co-location of size-3 or more, because it depends on the number of instances in each candidate co-location set. However, by using the sort-merge join strategy (the set of table instance is ordered by the class-id and instance-id), the complexity is lower (as the cost time of the inside of the iterating loop is less than the outside part in the experiment).

2.5 Experimental Evaluation

The performance of the algorithms with synthetic and real-world data sets has been evaluated. Synthetic datasets are generated using a methodology similar to the methodology used in Huang et al (2004). The synthetic data generator allows us to better control the study of the algorithms and the effects of interesting parameters.

The real-world plant distributed data set used in the experiments contains distribution information of plant species in the “Three Parallel Rivers of Yunnan Protected Areas”.

The experiments were performed on a Celeron computer with a 2.40 GHz CPU and 376 Mbytes memory running the Windows XP operating system. All programs are written in Java.

2.5.1 Performance Studying

It is considered that computing the fuzzy equivalence matrix is an important step in the algorithm. Therefore, two algorithms are evaluated in the performance study. One is the algorithm that uses the general method ($O(N^3)$) to compute the fuzzy equivalence matrix. It is denoted as GFCG (General Fuzzy Co-location Generation). The other is the OFCG (Optimized Fuzzy Co-location Generation). It uses an optimized method to compute the fuzzy equivalence matrix ($O(N^2)$).

The experiment is conducted using detailed simulations to answer the following questions:

(1). How does data density in the spatial framework affect the performance?

(2). How do the algorithms behave with different prevalence thresholds?

(3). How do the algorithms behave with difference level values λ of fuzzy equivalence matrices?

The common parameter values used in these experiments are as follows: the spatial framework is 250×250, the number of event types is fixed to 10, and the average area of a spatial instance is 25×25.

Effect of data density in the spatial framework: The effect of data density in the spatial framework was evaluated with spatial data sets generated by using above common parameters and spatial instances of different number, i.e., 100, 300, 500 and 700, to control the data density in spatial framework. Figure 2.7 shows the performance gain by two algorithms with the *min_prev* is set to 0.1 and the level value λ is equal to 0.1. As the density increases, the execution time of the two algorithms is dramatically increased. It shows the sensitivity of the algorithm with the increase of data density. For further investigation, Figure 2.8 shows a comparison between computing the fuzzy equivalence matrix (steps 2-4 in our algorithm) and the rest of the algorithm. As can be seen, the computation of fuzzy equivalence matrix consumes most of running time and demands a significant improvement. Table 2.2 is the results of mining from these synthetic data sets. It also gives a hint for what kinds of synthetic datasets are generated in these experiments.

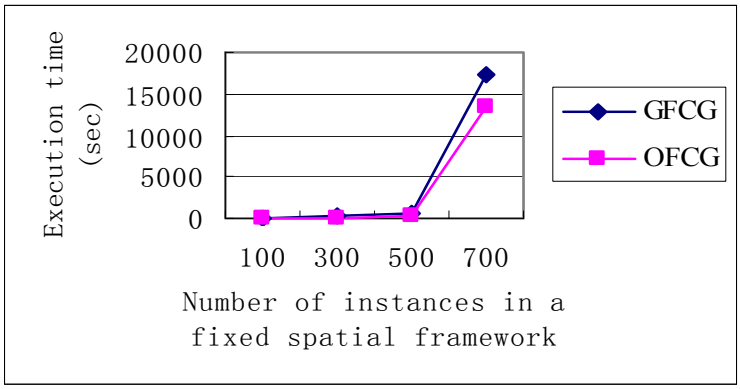


Figure 2.7 Effect of data density

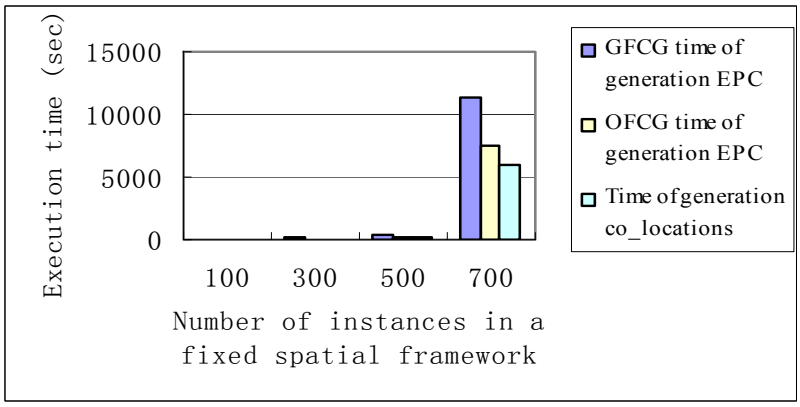


Figure 2.8 Comparison of density effect for the part of generation EPC and the rest of the part of the algorithm

Table 2.2 Mining results of synthetic data sets

Mining results / Data sets	Maximum size of co-location	Number of co-location
Dataset_100 instances	5	64
Dataset_300 instances	5	117
Dataset_500 instances	6	201
Dataset_700 instances	7	376

Effect of prevalence threshold: the performance effect, the prevalence threshold increases, sees Figure 2.9 (a). The experiment is conducted with the above dataset_300 instances, and λ is fixed to 0.2. It can be seen that the effect of the prevalence threshold for the two algorithms is almost the same. However, the execution time of algorithms decreases with the increase of the threshold value (Figure 2.9 (b). there is a blow up in Figure 2.9 (a)). The reason is the decrease in the number of joins of instances due to the efficient pruning. In addition, the execution time of the algorithm is much less than the first time, because steps 1-6 in the algorithm do not need to be run again.

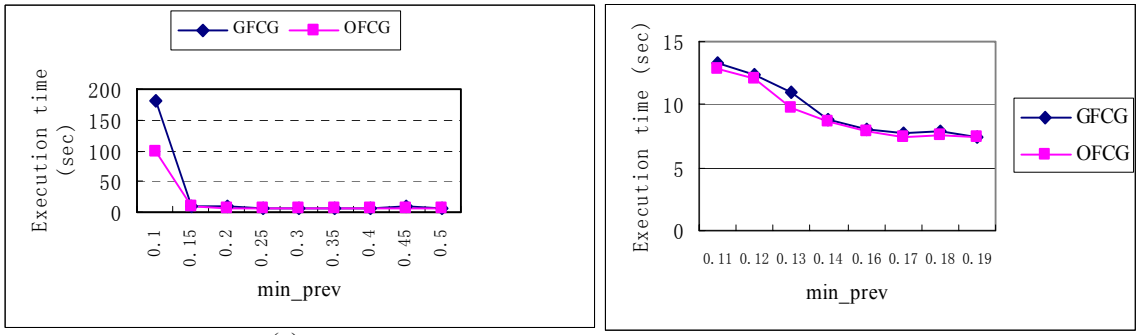


Figure 2.9 Effect of prevalence threshold

Effect of level value λ: The effect of level value threshold λ is evaluated with the synthetic dataset contained 300 instances, and *Min-prev* is fixed to 0.2. Figure 2.10 illustrates the execution time of the two algorithms as a function of the level values λ. When the level value λ is changed, the fuzzy similarity matrix and the fuzzy equivalence matrix do not need to be computed again in the two algorithms, so the execution time of the two algorithms is much less than the first time. Figure 2.10 (b) shows the execution time of the algorithm decreases, as the number of joins of instances decreases with the increase of the value of λ. The addition of class-id to the table does allow the set of table instances to be ordered by both class-id and instance-id.

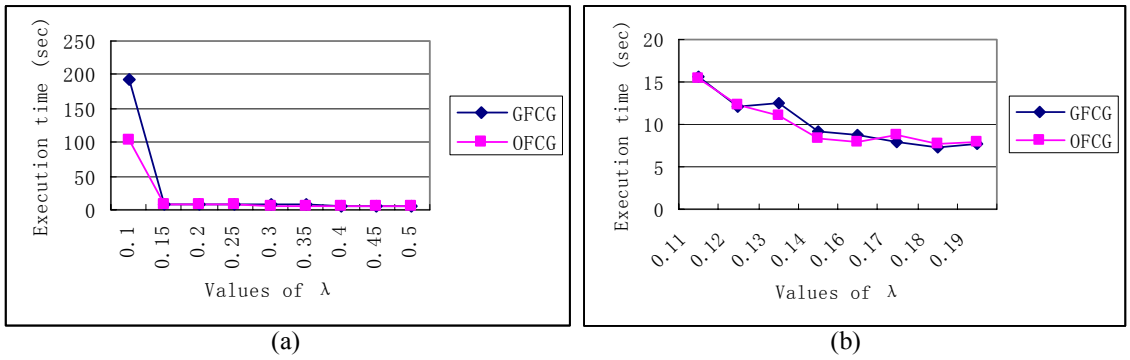


Figure 2.10 Effect of level value λ

2.5.2 Experiments on a Real Data Set

The algorithm is evaluated by using a plant distributed dataset of the “Three Parallel Rivers of Yunnan Protected Areas”. The number of plant species (event types) is 29. The total number of plant instances is 3908. When the level value λ is set to 0.09, the number of the fuzzy equivalence classifications is 252. When *Min_prev* is set to 0.1, the maximum size of co-location is 9 and the total number of size 2 co-location patterns is

167. Some selected mining results are shown in table 2.3. Table 2.4 is the corresponding table of plants' name and the ID used in table 2.3.

Table 2.3 Some selected results of mining fuzzy co-location on the "Three Parallel Rivers of Yunnan Protected Areas"

Rule_id	The left-hand side of the rules	The right-hand side of the rules	Cond_Prob*
1	(1)	(3)(7)(14)(15)	0.92
2	(1)(3)	(7)(14)(15)	0.97
3	(1)(7)	(3)(14)(15)	0.95
4	(1)(3)(7)	(14)(15)	1
5	(1)(3)(7)(14)	(15)	1
6	(2)	(4)(5)(9)	0.86
7	(2)(4)	(5)(9)	0.93
8	(2)(4)(5)	(9)	0.99
9	(3)	(2)(6)(8)(10)(11)(12)(13)	0.75
10	(3)(2)	(6)(8)(10)(11)(12)(13)	0.96
11	(16)	(17)(18)(19)(20)	0.80
12	(16)(17)	(18)(19)(20)	0.90
13	(16)(17)(18)	(19)(20)	1
14	(21)	(7)(11)(16)(22)(23)(24)	0.90
15	(21)(22)	(7)(11)(16)(23)(24)	1
16	(21)(22)(23)	(7)(11)(16)(23)(24)	1

* See section 2.2.2

Table 2.4 A correspondence table of plants' name and their ID used in table 2.3

Plants_id	Plants' name
(1)	Kobresia tunicata
(2)	Kobresia stiebritziana
(3)	Primula serratifolia
(4)	Vertrilla baillonii
(5)	Allium victorialis
(6)	Meconopsis venusta
(7)	Trollius yunnanensis
(8)	Juncus castaneus
(9)	Potentilla forrestii
(10)	Arenaria longistyla var. pleurogynoides
(11)	Anemone demissa
(12)	Chamaesium sp.
(13)	Polygonum coriaceum
(14)	Draba oreodoxa
(15)	Saxifraga macrostigma
(16)	Rhododendron cephalanthum
(17)	Salix calyculata
(18)	Kobresia prattii
(19)	Saxifraga sp.
(20)	Pedicularis sp.
(21)	Larix poteninii var. macrocarpa
(22)	Picea likiangensis
(23)	Schisandra rubriflora
(24)	Rubia sp.

By discussing with an expert botanist of Yunnan University (Zhiha Hu), it has been found from the mining results that there are three groups of co-location rules: The first group (rule 1 to rule 10 in table 3) represents a sub-class of a plant community, the second group (rule 11 to rule 13 in table 3) seems obvious as they belong to a plant community, and the third one (rule 14 to rule 16) is a mixed one of plant communities.

What is the information or knowledge behind the rules discovered by the algorithm? Prof. Hu explained it using Figure 2.11. The plants involved in rules 1-10 belong to a plant community, so they generally grow together. But if there is difference of terrain (for example, sloping field, steep valley or stone-swept terrain), the combination of a plant community represents that difference. The plants in rules 1-5 of table 3 are supposed to grow in a valley which is covered by aqueous soil and some big stones. The plants in rules 6-8 of table 3 are supposed to grow in a sloping field facing the West. The plants in rules 9-10 of table 3 are supposed to grow in a sloping area facing North-East.

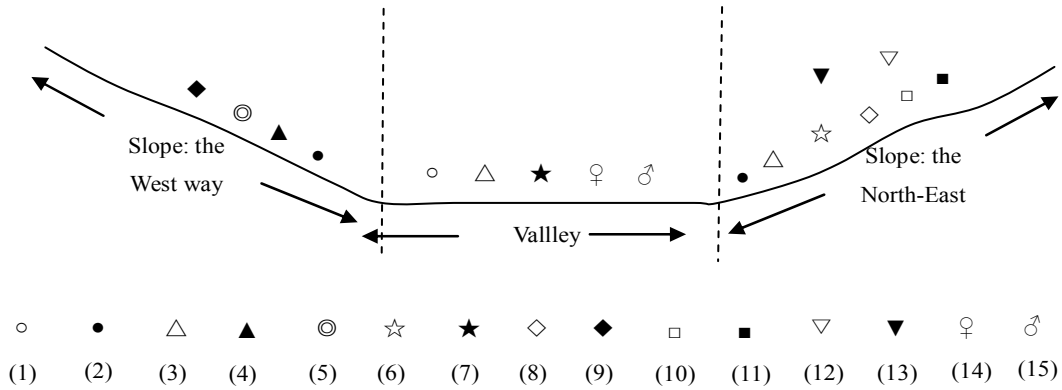


Figure. 2.11 An explanation of plants' distribution in fuzzy co-location patterns

In the second group of rules (rules 11-13 in table 3), plants (16) (17) belong to Bosks, and plants (18) - (20) belong to Herbages. They belong to a form, Rhododendron cephalanthum. Certain plants are expected to be co-located. Even if the threshold λ is set to a value which is small, it may be that the plants belonging to these communities are still not in a co-location pattern. This is because some problems might be with this area (or with these plant communities). For example, the ecological environment might be destroyed.

In the third group rules (rules 14-16 in table 3), plants (21) (22) belong to Arbors, plant (23) belongs to Frutexes, and plant (24) belongs to Herbages. They belong to the form, Larix pataninii var. macrocarpa. But why do plants (7) (11) (16) go together with this community, since they do not belong to the same form? Prof. Hu explained that those rules illustrate the plants (7) (11) (16) must be planted artificially (it is proved by the record's information of this area). It is a rare case in mining results.

We find that some plant species (for example, Plant "Cordyceps Sinensis (Berk.) Sacc") appear in many co-location rules. In fact, these plants grow in various places of this area. Our algorithm will be more efficient if these kinds of events are initialized in the set of events.

It is worth talking a few more words about the threshold λ which controls spatial neighbour relationship in fuzzy co-location mining. First, as we change the λ from low to high in experiments of the plants' dataset, the length of co-location patterns change from long to short. And plants in the same co-location pattern always belong to a plant community, even if the λ is very low. But in general co-location mining, the mining results are sometimes difficult to understand, as a co-location pattern might contain all of plants with the distance threshold *D* increased. Second, the fuzzy similarity matrix and the fuzzy equivalence matrix do not need to be computed again when the λ is changed, but the situation is not the same in general co-location mining.

2.6 Summary

This chapter analyzed the fuzzy characteristics of spatial events, and studied the problems of mining co-location patterns from fuzzy spatial datasets. Based on the concepts of the proximity between spatial instances and fuzzy equivalence partition, a semantic proximity neighbourhood controlled by the threshold λ was introduced. A prefix-based approach to partition the prevalent event set search space into subsets was presented, where each sub-problem can be solved in main-memory. In addition, the correctness, completeness, and computation cost of the algorithms were analyzed. An experimental study on a real plants' dataset showed that our proposed framework is effective for mining co-location patterns to identify the subsets of spatial events with significant spatial interactions.

In future work, how to obtain the fuzzy semantic proximity neighbourhoods and fuzzy participation index from the fuzzy matrix will be investigated. The effective use of the threshold λ in general co-location mining will also be explored. Furthermore, a plan to collaborate with domain experts to further investigate the fuzzy co-location patterns found in our experiment will be made. Finally, as well as studying fuzzy co-location mining, fuzzy spatial clusters, fuzzy spatial classification and fuzzy spatial outline detection on fuzzy spatial datasets will be investigated.

In next Chapter, a new join-less approach for co-location patterns mining will be presented. This work is very significant in the field of traditional spatial co-location patterns mining.

Chapter 3

A New Join-less Approach for Co-location Pattern Mining

With the rapid growth and extensive applications of the spatial dataset, it is becoming more important to solve how to find spatial knowledge automatically from spatial datasets. A spatial co-location pattern represents the subsets of spatial events whose instances are frequently located together in geographic space. It's difficult to discover co-location patterns because of the huge amount of data brought by the instances of spatial events. A large fraction of the computation time is devoted to generating the instances of co-location patterns. The essence of co-location patterns discovery and three kinds of co-location patterns mining algorithms proposed in recent years are analyzed in this chapter, and a new join-less approach for co-location patterns mining, based on the data structure----**CPI-tree** (Co-location Pattern Instance Tree), materializing spatial neighbour relationships, is proposed. All co-location table instances can be generated quickly with a CPI-tree. The correctness and completeness of the new approach are proved. Finally, an experimental evaluation using synthetic datasets and a real world dataset shows that the algorithm is computationally more efficient than the proposed algorithms.

3.1 Overview

Spatial data mining is the process to discover interesting and previously unknown, but potentially useful patterns from spatial datasets. Extracting interesting patterns from spatial datasets is more difficult than extracting the corresponding patterns from transaction datasets due to the complexity of spatial data types, spatial relationships and spatial autocorrelation (Han and Kamber, 2006; Agarwal and Srikant, 1994). A spatial co-location pattern represents a subset of spatial events which's instances are frequently located in a spatial neighbourhood. For example, botanists have found that there are orchids in 80% of the area where the middle-wetness green-broad-leaf forest grows. Spatial co-location patterns may yield important insights for many applications. For example, a mobile service provider may be interested in mobile service patterns frequently requested by geographical neighbouring users. The locations which are gotten together by people can be used for providing attractive location-sensitive advertisements, etc. Other application domains include Earth science, public health, biology, transportation, etc.

Co-location pattern discovery presents challenges due to the following reasons: First, it is difficult to find co-location patterns with traditional association rule mining algorithms since there is no concept of traditional "transaction" in most of spatial datasets (Agarwal and Srikant, 1994; Koperski and Han, 1995). Second, the instances of a spatial

event distribute in spatial framework and share complex spatial neighbourhood relationships with other spatial instances. So a large fraction of the computation time of mining co-location patterns is devoted to generating the table instances of co-location pattern.

3.1.1 Basic Concepts

Given a set of spatial events *F*, a set of their instances *S*, and a spatial neighbour relationship *R* over *S*. *R* could be topological relationships (e.g. linked, Intersection), distance relationships (e.g. Euclidean distance metric) and mixed relationships (e.g. the shortest distance of two points on a map). As shown in Figure 3.1, there are 4 spatial events *A, B, C* and *D* and their instances. *A.*1 stands for the first instance of *A*. If *R* is defined as a Euclidean distance metric and its threshold value is *d*, two spatial objects are neighbours if they satisfy the neighbour relationship: $R(\mathrm{A.1},\mathrm{B.1}) \Leftrightarrow (distance(\mathrm{A.1},\mathrm{B.1}) \le d)$. Given a subset of spatial instances $I = \{i_1, i_2, ..., i_m\}$, $I \subseteq S$. I is called as an **R-neighbour** if *I* forms a clique under the neighbour relation *R*.

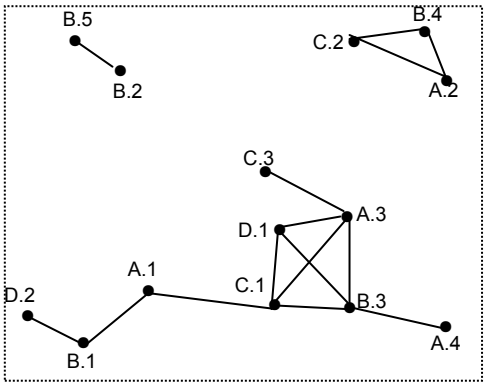


Figure 3.1 An example of spatial event instances

A **co-location** *c* is a subset of spatial events, i.e., $c \subseteq F$. An R-neighbour *I* is a **row instance** of a co-location *c* if *I* contains instances of all the events in *c* and no proper subset of it does so. The **table instance** of a co-location *c* is the collection of all row instances of *c*. The **size** of a co-location *c* is the number of spatial events in co-location *c*, it is denoted as $Size(c) = |c|$.

The interest degree of a co-location differs from the degree of support in traditional association rules mining. A new prevalence measure concept called the participation index is introduced by Huang, Shekhar and Xiong in (Huang et al, 2004). Participation ratio will be presented before giving the concept of participation index.

The **participation ratio** $PR(c, f_i)$ for event type f_i in a size-k co-location *c*= $\{f_1 ... f_k\}$ is the fraction of instances of event f_i which participate in any row instance of co-location *c*. The participation ratio can be computed as $PR(c, f_i) = \frac{\pi_{f_i}(|table\_instance(c)|)}{|table\_instance(f_i)|}$, where π is the relational projection operation with duplication elimination.

The **participation index** of a co-location $c=\{f_1 \ldots f_k\}$ is the minimum in all $PR(c, f_i)$ of co-location *c*:

$$PI(c)=\min_{i=1}^{k}\{PR(c, f_i)\}。$$

Example 3.1. Take Figure 3.1 as an example. *A* has 4 instances, *B* has 5 instances, *C* has 3 instances, and *D* has 2 instances. Suppose co-location $c=\{A,B,C\}$, the table instance of co-location *c* is $\{\{A.2,B.4,C.2\},\{A.3,B.3,C.1\}\}$. $PR(c,A)$=2/4 since there are only *A.2* and *A.3* in this table instance. Similarly, $PR(c,B)$ =2/5, $PR(c,C)$ =2/3. $PI(C)=\min(PR(c,A),PR(c,B),PR(c,C))=2/5$.

Given a minimum prevalence threshold *min_prev*, a co-location c is a **prevalent co-location** if $PI(c)\geq$ min_prev holds.

Lemma 3.1. The participation ratio and the participation index are monotonically non-increasing with the size of the co-location increasing.

Proof: Suppose a spatial instance is included in the row instance of co-location *c*. For co-location $c'\subseteq c$, this spatial instance must be included in the row instance of c'. The opposite is not true. Therefore, the participation ratio is monotonically non-increasing.

Suppose $c=\{e_1,\cdots,e_k\}$,

$$Pi(c\cup e_{k+1})=\min_{i=1}^{k+1}\{\Pr(c\cup e_{k+1},e_i)\}\leq\min_{i=1}^{k}\{\Pr(c\cup e_{k+1},e_i)\}\leq\min_{i=1}^{k}\{\Pr(c,e_i)\}=Pi(c)$$

Therefore, the participation index of co-location is also monotonically non-increasing.

Lemma 3.1 ensures that the participation index can be used to effectively prune the search space of co-location pattern mining.

3.1.2 Problem Definition

The co-location mining problem is formalized as follows:

Given:

1) A spatial framework η

2) A set of spatial events $F=\{f_1,\ldots,f_n\}$ and a set of their instances $S=S_1\cup S_2\cup\ldots\cup S_n$, $S_i(1<i<n)$ is the set of instances of the event f_i, and each instance is a vector <feature type, instance id, location>, where location$\in\eta$.

3) A spatial neighbour relation *R* over *S.*

4) A minimum prevalence threshold *min_prev*

Find:

A set of co-location patterns with participation index $\geq$ *min_prev*

3.1.3 Background for Mining Co-location Patterns

In previous work on mining co-location patterns, Morimoto (2001) defined distance-based patterns called k-neighbouring class sets. In his work, the number of instances for each pattern is used as the prevalence measure, which does not possess an anti-monotone property by nature. However, Morimoto used a non overlapping instance constraint to get the anti-monotone property for this measure. In contrast, Shekhar and Huang (2001) developed an event centric model, which does away with the non-overlapping instance constraint. They also defined a new prevalence measure called the participation index. This measure possesses the desirable anti-monotone property. The related work in the approach proposed by Shekhar and Huang can be classified into three kinds for identifying co-location instances: the full-join approach, the partial-join approach and the join-less approach.

(1). The full-join approaches: The full-join approach is mainly based on the computation of the join operation between table instances for identifying co-location instances. First, the spatial neighbour relations between spatial instances are found out, and all the tables instance of size-2 co-location pattern are generated; second, generate the size-3 co-location table instances by joining size-2 table instances. Size-k+1 co-location table instances are generated by joining size-k co-location table instances. This approach is similar to *Apriori* method and it could generate correct and complete co-location sets. However, scaling the algorithm to substantially large dense spatial datasets is challenging due to the increasing number of co-locations and their table instances.

(2). The partial-join approaches: The algorithm proposed by Yoo and Shekhar in (Yoo and Shekhar, 2004) is to build a set of disjoint cliques in spatial instances to identify the intraX instances of co-location (belonging to a clique) and interX instances of co-location (belonging to between two cliques), and join the intraX instances and interX instances respectively to calculate the value of PI (The participation index). This approach reduces the number of expensive join operations dramatically in finding co-location instances. However, the key of this algorithm is to find cliques as big as possible, which could cut down the spatial neighbour relationships between two cliques. Besides building cliques is time-consuming, if the correct cliques could not be identified, and the number of cut neighbour relations would not be decreased, the partial-join algorithm of mining co-location pattern would be similar to the full-join algorithm.

(3). The join-less approaches: The algorithm proposed by Yoo, Shekhar and Celik in (Yoo et al, 2005) puts the spatial neighbour relationships between instances into a compressed *star neighbourhood*. All the possible table instances for every co-location

were generated by scanning the star neighbourhood, and by 3-time filtering operations. The join-less co-location mining algorithm is efficient since it uses an instance-lookup scheme instead of an expensive spatial or instance join operation for identifying co-location table instances. However, the star neighbourhood structure is not an ideal structure for generating table instances, for the table instances generating from this structure have to be filtered. Therefore, the computation time of generating co-location table instances will increase with the growing of length of co-location patterns.

3.1.4 Motivation

Let us see the spatial instances in Figure 3.1. If a pair of spatial instances satisfy neighbour relationship R, connect them with a solid line (as seen in Figure 3.1), then a graph G (E, V) can be obtained. Each co-location instance is a complete graph (clique) in G. Mining co-location patterns is equal to the process of mining all cliques in G and calculating the *PI* value of each co-location pattern. However, such process has been proved as a *NP-Hard* problem (Alsuwaiyel, 2004). In fact, in the process of discovering cliques, according to the definition of co-location pattern, the same spatial events cannot appear in a clique, and according to the anti-monotonic property of *PI* value (Lemma 3.1), not all the cliques should be calculated. The available 3-kind of mining co-location pattern algorithms is based on the two properties and similar to *Apriori* approach.

Can all the cliques be calculated through simply scanning G? Can a structure which contains the information of co-location table instances be built? In this Chapter, a new structure called CPI-tree (Co-location Pattern Instance Tree), which is similar to FP-tree, will be introduced in here. It could materialize the neighbour relationships of a spatial data set, and find all the co-location table instances recursively from it. Different from the star neighbourhood structure in the join-less approach, all information of the neighbour relationships in a spatial dataset is organized together by the CPI-tree. So, the third phase filters in the join-less algorithm, which might be an expensive step, need not be performed. Meanwhile, some filtering methods appeared in the join-less algorithm, which can filter candidate co-locations without finding exact co-location instances, will be reserved. Some new filtering methods will be considered in the new algorithm. As it happens, in the CPI-tree-based approach, all co-location table instances are discovered by scanning the CPI-tree once without candidate co-location generation. Although, in many cases the *Apriori* candidate generate-test method reduces the size of candidate sets significantly and leads to performance gain. However, it may need to repeatedly scan the star neighbourhood and check a large set of candidates by pattern matching. This is especially the case for mining long patterns.

3.1.5 Organization of the Chapter

The remainder of the Chapter is organized as follows. The *CPI-tree* structure and the method to construct it are introduced in Section 3.2. Section 3.3 develops a CPI-tree-based complete co-location-instance generating algorithm. Section 3.4 explores some pruning strategies and optimized strategies for improving efficiency of the CPI-tree-based algorithm. The experimental results are presented in Section 3.5. Section 3.6 summarizes the study and points out some future research work.

3.2 Co-location-Instance Tree (CPI-tree): Design and Construction

In this Section, the structure of *CPI-tree* and its construction method are introduced.

3.2.1 CPI-tree

A compact data structure can be designed based on the following observations:

(1). Since spatial neighbour relations between two instances make certain all co-location-instance, it is necessary to perform one scan of spatial datasets to identify the set of neighbour relationships.

(2). If the set of neighbour relationships can be stored in a compact data structure, it may be possible to avoid repeatedly scanning the set of neighbour relationships (the *apriori-like* algorithms did so, because they have to scan all size-k co-location-instance when they generating size-k+1 co-location-instance.)

(3). If a size-k co-location-instance is found, it may be cost-efficient to expand a size-k+1 co-location-instance from it at once. It is easy to expand co-location-instance if the set of neighbour relationships can be stored in a compact data structure.

(4). The recursive and hierarchical properties of the tree structure ensure the clarity and simplicity of the algorithms' description. If all spatial instances are sorted in ascending order (the spatial events in alphabetic order, and then the different instance of the same spatial event in numerical order), a graph G representing spatial neighbour relationships may correspond to a unique tree structure.

With the above observations, a tree structure (called CPI-tree (Co-location Pattern Instance Tree), for all co-location-instances can be generated from it) can be defined as follows.

Definition 3.1 (CPI-tree). A CPI-tree is a kind of rooted tree. The root of CPI-tree is labelled as "null". A branch of the CPI-tree is constructed a corresponding connective sub-graph in the graph G. The nodes in CPI-tree represent the spatial instance. The node *u* is the parent of the node *v*, when there is a neighbour relationship between instances *u* and *v* and the instance *u* is "smaller" than instance *v*.

Based on this definition, there is the following CPI-tree construction approach.

1) Create the root of a CPI-tree, and label it as “null”.

2) Push all the spatial instances into a stack T1 in alphabetic and numerical descending order.

3) Pop an instance from the stack T1, create a child node of the root “null” for this instance (e.g. A.1 in Figure 3.3). Push this instance into a stack T2.

4) Pop an instance (e.g. A.1) from stack T2. Find out all the instances which are the neighbours of this instance (showed in Figure 3.2), the different spatial event from this instance, and “bigger” than this instance. These instances form child nodes of this node in ascending order. Delete all these instances from stack T1 and stack T2, and link to the same instance-name node (except for leaf-nodes) in the CPI-tree (see dashed link in Figure 3.3).

5) Push child node instances (they have not the same instance-name node in CPI-tree) into the stack T2 in descending order. Then, turn to 4).

6) Repeat the operation above till the stack T2 is empty, and then turn to 3)

7) Repeat the operation above till the stack T1 is empty.

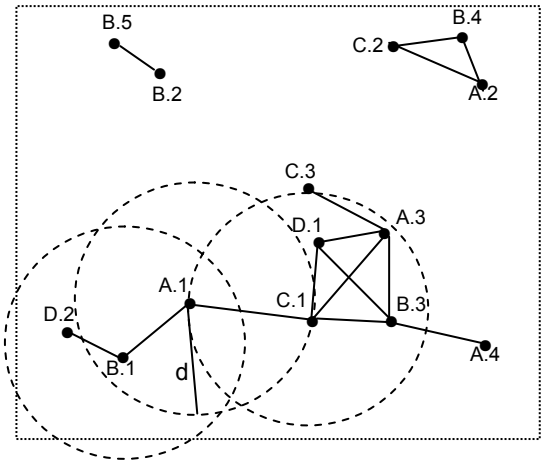


Figure 3.2 Neighbours of the instance A.1, B.1 and C.1.

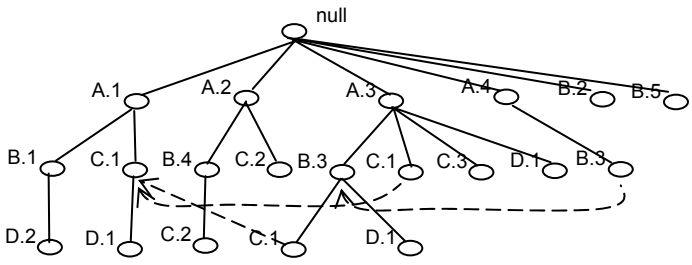


Figure 3.3 CPI-tree of the example in figure 3.1

The Figure 3.3 is the CPI-tree of the example in Figure 3.1. The CPI-tree of a spatial dataset constructed by above steps will be unique. The CPI-tree materializes the

neighbour relationships of a spatial dataset with no duplication of the neighbour relationships and no loss of co-location instances, and more important thing is that it is convenient and efficient to generate the co-location instances from it.

The approach of constructing a CPI-tree can be converted into the following algorithm.

Algorithm 3.1 (CPI-tree construction).

Input: *S*: a set of spatial instances and each instance is a vector <feature type, instance id, location>; *R*: the spatial neighbour relationship (e.g. Euclidean Distance);

Variables: *NT*= $\{NT_{l_1}, NT_{l_2}...NT_{l_m}\}$: a set of spatial neighbour relationships where NT_{l_i} is the set of neighbour instances of the instance l_i, whose order is "bigger" than l_i, and is sorted in ascending order of instances.

Output: *CPI-tree*: The CPI-tree structure of materialized spatial neighbour relationships;

Method:

1) *NT=gen_neighbourhood;*
2) Create the root of a CPI-Tree, and label it as "null."
3) Push all the instances in *S* into a stack T1 in descending order;
4) **While** the stack T1 is not empty **Do**
5) { *I* ← Pop an instance from stack T1;
6) create the node *I* which is the child of the root "null";
7) Push instance *I* into the stack T2;
8) **While** the stack T2 is not empty **Do**
9) { *I* ←Pop an instance from the stack T2;
10) **For** each instance *f* in NT_I **Do**
11) { Create the node *f* which is the child of the node *I*;
12) Delete the instance *f* from the stack T1 and T2;
13) Link the node *f* to the same instance-name non-leaf-node in CPI-tree;
14) If there is not the same instance-name with the node *f* in CPI-tree then *f* → NT_I' }
15) *Push* all the instances in NT_I' into the stack T2 in descending order;
}
}
16) Return the root "null"

The procedure *gen_neighbourhood* of this algorithm generates a set *NT*= $\{NT_{l_1}, NT_{l_2}...NT_{l_m}\}$ of spatial neighbour relationships, where NT_{l_i} is the set of neighbour instances of the instance l_i whose order is "bigger" than l_i and is sorted in ascending order of instances. In this procedure, all neighbour instance pairs are found firstly according to spatial neighbour relationship *R*, and then the set

$NT = \{NT_{l_1}, NT_{l_2} ... NT_{l_m}\}$ is generated by grouping the neighbour instances for each instance.

In the loop of the step 4), each of which will create a branch of the root "null" in the *CPI-tree*, and the loop of the step 8) generates iteratively all nodes of this branch of the CPI-tree. As it happens, the instances are considered in ascending order in the step 10), because the instances are sorted in ascending order in NT_l. The operation deletion in the step 12) and the operation of the step 14) are to avoid duplication of information, and the operation linking in the step 13) is for no loss of information.

3.2.2 Complexity and Completeness of CPI-tree

Analysis. The computational complexity of the algorithm 3.1 includes procedure Gen_neighbourhood and the rest of algorithm. Suppose the number of spatial instances is *m*. In the worse case, the computational complexity of the procedure Gen_neighbourhood will be $O(m^2 \log_2 m)$, and the rest of the algorithm will be $O(m^2)$. It will be shown that the CPI-tree contains the complete and no redundant information for table-instance generating in Lemmas 3.2.

Lemma 3.2. All the neighbour relationships of given spatial instances are recorded in a CPI-tree, no one is omitted.

Proof: according to the procedure of constructing a CPI-tree, all the spatial instances are scanned and their neighbour relationships are recorded in CPI-tree. Therefore, none of the spatial instances' neighbour relationship is omitted in CPI-tree.

Lemma 3.3. CPI-tree materializes the neighbour relationships of a spatial dataset with no duplication of the neighbour relationships.

Proof: It is obvious because the step 12) in constructing a CPI-tree algorithm guarantees each spatial neighbour relationship is considered once, and the step 10) and step 14) ensure that a connective sub-graph in spatial dataset forms a branch of a CPI-tree.

3.3 Generating Complete Table-Instance Using *CPI-tree*

Construction of a CPI-tree which materializes the neighbour relationships of a spatial dataset ensures that all table instances can be generated from this CPI-tree. However, it is not guaranteed that the process of generating table instances will be highly efficient, since one may still encounter the combinatorial problem of table instances generation if one simple uses this CPI-tree to generate all table instances.

In this section, it will be studied that how to scan the neighbour relationship stored in a CPI-tree, develop the principles of generating table instances using CPI-tree, explore

how to perform further optimization of generating table instances using CPI-tree, and propose a table-instance generation algorithm, *Gen_instance*, for efficiently generating the complete set of table instances using CPI-tree.

3.3.1 Principles of Table-Instance Generation from a CPI-tree

In this subsection, some interesting properties of the CPI-tree structure which will facilitate co-location-instance generation will be examined. To facilitate properties description, data structure of node in CPI-tree is defined as follows. Each node in the CPI-tree consists of three fields: instance-name, child-link, and node-link. Node-link links to the non-leaf node in the CPI-tree carrying the same instance-name, or null if there is none.

Definition 3.2 (*direct-child-link*). A link between two nodes in a CPI-tree is called as a *direct-child-link*. A k-length direct-child-link is called as a *size-k direct-child-link*. The nodes lying in middle of a size-k direct-child-link are called as *intra-node*. For a node in a direct-child-link, the nodes lying in this node below are called as *child-node* on the direct-child-link.

Example 3.2 Considering examples in Figure 3.3, “A.3 B.3” is a direct-child-link, “A.3 B.3 C.1” is a size-3 direct-child-link, B.3 is the intra-node of the size-3 direct-child-link “A.3 B.3 C.1”, and B.3 and C.1 are the child-nodes of A.3.

Definition 3.3 (*indirect-child-link*). The child nodes linked out by a node-link (i.e., dashed link) are called as *indirect-child.* A size-k child-link linked out by an indirect-child is called as a size-k *indirect-child-link*.

Example 3.3 D.1 is the *indirect-child* of the node C.1 linked to its parent by the node-link in Figure 3.3. So, “B.3, C.1, D.1” and “A.3, C.1, D.1” are size-3 indirect-child-link.

Definition 3.4 (*all-link*). If all child-nodes of each intra-node in a size-k direct-child-link are the brothers of corresponding intra-node, then the size-k direct-child-link is called as a size-k *all-link*. If the size-k direct-child-link defined above is a size-k indirect-child-link, then it is called as a size-k *indirect-all-link*.

Example 3.4 In Figure 3.3, the size-4 direct-child-link “A.3 B.3 C.1 D.1” is a size-4 indirect-all-link. Because C.1 and D.1 are the brothers of B.3, and D.1 is the brother of the node C.1. In the same way, the size-3 direct-child-link “A.3 B.3 D.1” is a size-3 direct-all-link.

Based on the definitions above, there are the following properties.

Property 3.1 (*Child-link property*). Each size-2 child-link in a CPI-tree denotes a size-2 co-location instances.

This property can be obtained directly from the CPI-tree constructing process, and it is the base of generating other co-location instances.

Property 3.2 (*All-link property*). Each size-k all-link or size-k indirect-all-link in a CPI-tree denotes a size-k co-location table instance. (k>2)

Rationale. It is obvious that the instances satisfying all-link or indirect-all-link form a clique in corresponding graph G.

Example 3.5 The size-3 all-link "A.3 B.3 D.1" forms a co-location instance "A.3, B.3, D.1", the same to the size-4 indirect-all-link "A.3 B.3 C.1 D.1" and size-3 indirect-all-link "B.3 C.1 D.1".

It then will be shown that the complete set of table instances in a spatial dataset can be generated by using corresponding CPI-tree.

Lemma 3.4 Co-location instances generated from the CPI-tree by using **properties 3.1** and **3.2** are correct and complete.

Rationale. First, it is shown that each table instance generated from properties 3.1 and 3.2 is correct and distinct. The correctness is guaranteed by **property 3.1** and **property 3.2**, and the distinction is guaranteed by **lemma 3.3**.

Second, it is shown that no co-location instance can be generated out of the CPI-tree. Suppose a size-k co-location instance can be generated out of the CPI-tree. If this is a size-2 co-location instance, and then there is not a child-link between the two instances. According to **lemma 3.2**, there is not a spatial neighbour relationship between the two instances, this reduces to absurdity. For size-k co-location instance (k>2), that means that these instances presenting to the size-k co-location instance do not form a all-link or indirect-all-link, and then according to the process of constructing a CPI-tree, there at least is not a neighbour relationship between two instances among these instances. This also reduces to absurdity.

3.3.2 Table-Instance Generation Algorithm

Complete co-location table instances of a spatial dataset are generated recursively from corresponding CPI-tree in the following steps.

1) Initiate prefix pattern α= {null} (the root of CPI-tree) and the suffix pattern β= {null}, search CPI-tree recursively for complete the co-location table instances.

2) If α has a child node *s* and *s* has a child node *t* too, then α is put by *s*, i.e. α=*{s}*, search CPI-tree recursively with the new α and β; else (including the last recursion returns), (1). β is put by *s*, i.e. β=*{s}*, and a co-location instance $\alpha \cup \beta$ is generated; (2). if β

has child node *t*, then (a). Generating a size-3 co-location instances if the $\alpha \cup \beta \cup \{t\}$ is a *all-link* (i.e., the child node *t* of β is the brother node of β), examining the size-4 co-location instances if *t* has child node…, … (b). If β has indirect-child nodes, examining indirect-all-link to generate some new co-location instances. For example, consider the node "B.3" in Figure 3.3, when β is "B.3", besides generating the instances "A.3, B.3, C.1" and "A.3, B.3, D.1", the instance "A.3, B.3, C.1, D.1" also be generated.

3) If the recursion is ended, return all the co-location table instances.

Example 3.6. Generating all co-location table instances of the CPI-tree in Figure 3.3

First level recursion (α=null, β=null): because the child node A.1 of α has children, go to the second level recursion.

Second level recursion (α= {A.1}, β=null): because the child node B.1 of α still has children, go to the third level recursion.

Third level recursion (α={B.1}, β=null): because the child node D.2 of α has no child, so β={D.2}, a table instance of co-location *{B, D}* is $\alpha \cup \beta$ = "B.1, D.2". The D.2 is a leaf node, end this recursion. When returning to the A.1, first, "A.1, B.1" forms a co-location instance. Second, examine "A.1 B.1 D.2" is an all-link or not for determining it is a new co-location instance or not (here, it is not). After finishing this branch, go to another branch of A.1. A.1 has another child C.1, go to the next recursion level…, until to a leaf node (when come to the node D.1), the table instance of co-location pattern {C, D} "C.1, D.1"will be generated, and then return to the upper recursion level…. After the table instance "A.1,C.1" of pattern {A, C} is generated, the branch A.1 finish due to the "A.1, C.1, D.1" is not an all-link…….

The above approach can be transformed into the following algorithm.

Algorithm 3.2 (Gen_instance).

Input: *CPI-tree*: materialized spatial neighbour relationships in a spatial dataset and constructed according to Algorithm 3.1;

Output: The complete set of co-location table instances.

Method: call Gen_instance (CPI-tree, null, null)

Procedure Gen_instance (*CPT-Tree*，α，β)

1） **while** α has child node *s* **Do**

2） {***if*** *s* has child node *t* **then** Gen_instance (CPI-Tree, *{s}*, β);

3） β= *{s};*

4） If α <> *"null"* **then**

5） **{** $\alpha \cup \beta$ forms an instance of co-location pattern;

6） ***If*** β has a child-node or a indirect-child-node **then** *Gen_next_instance (CPI-tree, α, β)*; }

7） }

Procedure *Gen_next_instance (CPI-tree, α, β)*

1） { ω= The last element of β;

2） **while** (ω has a child-node or a indirect-child-node *t* AND *Ok* (t, β) **Do**

3） { β=$\beta \cup \{t\}$;

4） $\alpha \cup \beta$ forms an instance of co-location pattern;

5） *If t* is not a leaf **then** *Gen_next_instance (CPI-tree, α, β) }*

6） }.

Bool Function *Ok* (t, β) //If the node t is the brother of β, then return true, else return false

1） { **For** i=1 to |β|

2） **If** the node *t* is not the brother of the i-th element in β **then** Return false;

3） Return true }

Analysis. With the properties and lemmas in Section 3.2 and 3.3, the algorithm correctly finds the complete set of table instances in a spatial dataset.

From the algorithm and reasoning, one can see that the table instances generating process is a backtracking process. Let's now examine the efficiency of the algorithm. The algorithm scans the CPI-tree once and table instance generating is recursively performed on the child-link from size 2. If a size-k table instance is not generated, then the size-k+1 table instance derived from it will not be considered. Moreover, after generating lower table instances, the table instances derived from them will be examined at once. This is much less costly than traditional operation method of the full-join and the join-less. Thus the algorithm is efficient. The real execution results will be shown in Section 3.5.

3.4. Some Optimization Strategies

3.4.1 Pruning Strategies

Although generating co-location instances from a CPI-tree will be No loss of co-location instances and no duplication of co-location instances, the following pruning strategies can be used to improve efficiency of generating co-location instances from CPI-tree.

Pruning 3.1. A node, which is the child of the root "null" and has no child, can be pruned.

Proof. If a node is the child of the root "null" and it has not a child node, it must be the spatial instance without neighbourhood. So it can be pruned.

Example 3.7 In Figure 3.3, the instances B.2 and B.5 can be pruned with Pruning 1.

Pruning 3.2. By using **Pruning 3.1**, If the number of the pruned instances of a feature f_i is greater than *min_prev**| f_i|, then all the instances of the feature f_i and the relevant edges in the CPI-tree can be pruned. (If the root node of the remaining branch is the right brother node of the pruned node, move the remaining branch to the right brother node and combine the same nodes; else, move the remaining branches to become another branch of this CPI-tree root.)

Proof. If the number of the pruned spatial instances of a feature f_i with **Pruning 3.1** is greater than *min_prev**| f_i|, the number of the remaining instances of the feature is less than the *min_prev**| f_i|. Therefore, all instances of this spatial feature might be pruned due to the co-location containing the feature might not be prevalent.

Example 3.8 Suppose that three instances of spatial feature B was pruned with Pruning 3.1, and there were five instances in feature B and the *min_prev* is 50%, then all the instances of B and the relevant edges can be pruned with **Pruning 3.2**.

3.4.2 Optimization by Reducing the Depth of CPI-tree

In the algorithm Gen-instance, an all-link forms a co-location instance. One can see that cutting a branch in a CPI-tree that may not form an all-link will not affect the results of the algorithm Gen-instance. For example, the instance “B.3” in Figure 3.4 (a), for its children may not form any all-link with her parent, the branch “B.3” can be cut (duplicate exactly) and move it to the root “null” of this CPI-tree (see Figure 3.4 (b)). The same with the branch “B.1” is. For some distribution of data sets, spatial instances are connected by some links (they can be called as bridges). Because of these bridges, the instances, they could not be generated a co-location instance, are connected together, that must affect efficiency of the algorithm. So, these links are cut for optimizing the method.

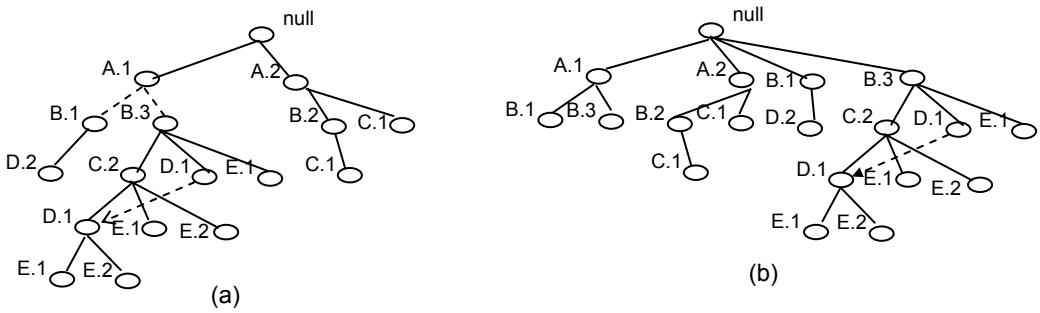


Figure 3.4 An Example of reducing the depth of CPI-tree

Definition 3.5 (*Cut-link*) A link connecting the node i_1 to i_2 in a CPI-tree is called a *cut-link* if any child-node of i_2 is not its brother.

Figure 3.4(a) presents cut-link as dotted lines. Link {A.1, B.1} and {A.1, B.3} are cut-links.

Property 3.3 (*Cut-link property*) The child instances linked out by a cut-link cannot form a co-location instance with its parent instance.

Rationale. First, the property 3.3 guarantees there is not any size-3 co-location instance linked out by a cut-link. Second, there is not any size-k (k>3) co-location instance linked out by this cut-link according to Property 3.2.

Therefore, for a cut-link $\{i_1, i_2\}$, moving branch i_2 to the root "null" will not affect the results of generating co-location instances from CPI-tree (see Figure 3.4(b)). There is a duplication of the node "i_2" is generated in this optimization process, for generating size-2 co-location instance $\{i_1, i_2\}$.

It is obvious that the co-location instances generating from the depth-reduced CPI-tree is better than from the original CPI-tree, but when and how the process of reducing the depth of a CPI-tree is performed. For considering the efficiency of this process, it is done when the CPI-tree is building. It means that the process of constructing a CPI-tree will be changed for considering optimization strategies of reducing the depth of a CPI-tree.

Algorithm 3.3 (construction of a CPI-tree for considering optimization strategies of reducing its depth)

Method:

1） *NT=gen_neighbourhood;*
2） Create the root of a CPI-Tree, and label it as "null."
3） Push all the instances in *S* into a stack T1 in descending order；
4） **While** the stack T1 is not empty **Do**
5） **{** *I*← Pop an instance from stack T1；
6） create the node *I* which is the child of the root "null"；
7） Push instance *I* into the stack T2；
8） **While** the stack T2 is not empty **Do**
9） **{** *I*←Pop an instance from the stack T2;
10） **For** each instance *f* in NT_I **Do**
11） **{** Create the node *f* which is the child of the node *I*；
12） Delete the instance *f* from the stack T1 and T2；
13） Link the node *f* to the same instance-name non-leaf-node in CPI-tree;
14） If there is not the same instance-name with the node *f* in CPI-tree then $f \rightarrow NT_I'$ **}**
15） *Push* all the instances in NT_I' into the stack T2 in descending order;
16） **If** all the instances in NT_I are not the brothers of the instance *I* **then**
17） copy the node *I* and move the branch *I* to the root "null"; //branch *I* includes the node *I* and its children//
}

}

18）Return the root "*null*"

Comparing to **algorithm 3.1**, besides the steps 16) and 17) are added, the others are all the same. So, when using **algorithm 3.3** to build a CPI-tree, it can be seen that a depth-reduced CPI-tree is obtained. The experiments in Section 3.5 use this optimized algorithm.

3.5. Experimental Results

In this section, the performance of the algorithms is evaluated with the join-less approach using both synthetic and real data sets. All the experiments were performed on a 3-GHz Pentium PC machine with 2G megabytes main memory, running on Microsoft Windows/XP. All programs are written in Java.

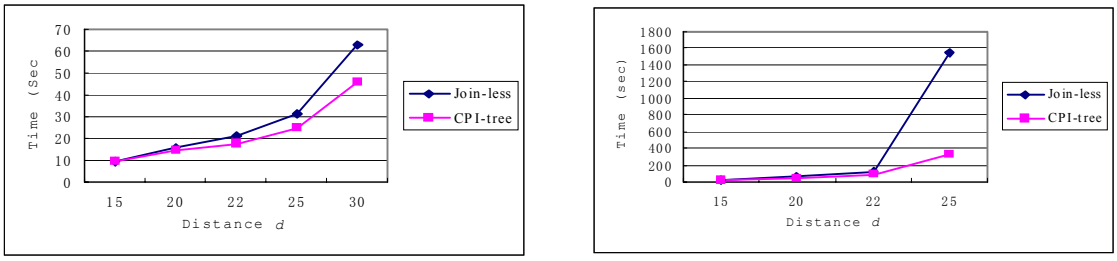
The experimental results are reported on two synthetic data sets. The first one is called as sparse dataset with 26 spatial features. In this dataset, when the neighbour distance threshold *d* and the prevalence threshold *min_prev* are set to 25 and 0.15, the total number of size 2 co-locations and the maximum size of co-locations are 104 and 4, respectively, while the number of all instances in the dataset is set to 10k. The prevalent co-locations are short and not numerous in this dataset.

The second synthetic dataset used in the experiments is a dense dataset with 26 spatial features. The total number of size 2 co-locations and the maximum size of co-locations are 232 and 8, when the threshold *d* and the *min_prev* are set to 25 and 0.15, respectively. There exist long prevalence co-locations as well as a large number of short prevalence co-locations in this dataset when the prevalence threshold *Min_prev* goes down.

To test the practicability of CPI-tree, a real dataset, the plant distributed data set of the "Three Parallel Rivers of Yunnan Protected Areas", is used. It contains the number of plant species (spatial event types) is 16. The total number of plant instances is 3908. When *Min_prev* and distance *d* are set to 0.1 and 1900 respectively, the maximum size of co-location is 4 and the total number of size 2 co-location patterns is 42. The characteristic of the dataset is that there are a large number of table instances in each co-location pattern.

1） Scalability with the neighbour distance *threshold d* over sparse data set and dense data set: The runtime of *CPI-tree* and *Join-less* on the sparse and the dense synthetic datasets, when the prevalence threshold *min_prev* is set as 0.5, as the

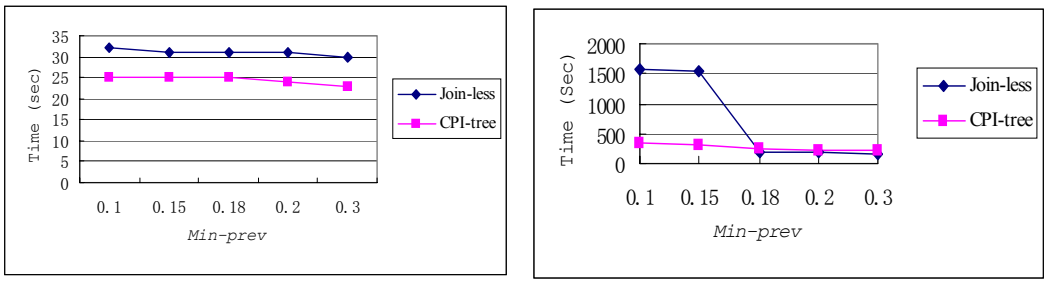
neighbour distance threshold *d* increases from 15 to 25/30 is shown in Figure 3.5 (a), (b). Since the dataset is sparse, as the threshold *d* is low, the prevalent co-location patterns are short and the set of such patterns is not large, the advantages of CPI-tree over Join-less are not so impressive. However, as the threshold *d* goes up or the dataset becomes dense, the gap becomes wider.



(a). over sparse data set (b). over dense data set

Figure 3.5 Scalability with *distance d* over sparse data set and dense data set

2）Scalability with prevalence threshold *Min_prev* over sparse data set and dense data set: Figure 3.6(a) shows the experimental results of scalability with *Min_prev* over the sparse dataset, while the results over dense dataset are shown in Figure 3.6(b). The neighbour distance threshold *d* is set as 200 in the experiments of Figure 3.6(a), while *d* is 150 in the experiments of Figure 3.6(b). The advantage of CPI-tree approach is more impressive with threshold *Min_prev* decrease and the dataset becomes dense.



(a). over sparse data set (b). over dense data set

Figure 3.6 Scalability with *Min_prev* over sparse data set and dense data set

3）Scalability with prevalence threshold *Distance d* over a real data set: The mining result over a real dataset, a plant distributed data set of the “Three Parallel Rivers of Yunnan Protected Areas”, is shown in Figure 3.7. From the figure, one can see that CPI-tree method is scalable even when there are many table instances. In such real datasets, the join-less method is not comparable to the performance of CPI-tree method.

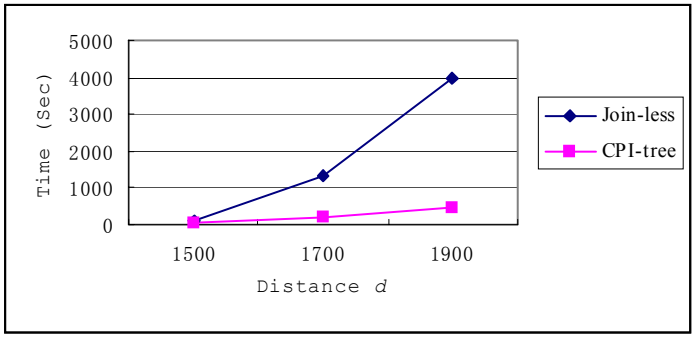


Figure 3.7 Scalability with *Distance d* over a plant distributed data set of the "Three Parallel Rivers of Yunnan Protected Areas"

4）Scalability of CPI-tree algorithm with number of instances: To test the scalability of CPI-tree against the number of instances, the dense dataset is used with *Min-Prev* is set to *0.3,* the neighbour distance threshold *d* is 20, and the number of instances ranges from 3K to 15K. The result is shown in Figure 3.8, which shows that the CPI-tree method is the linear increase of runtime with the number of instances.

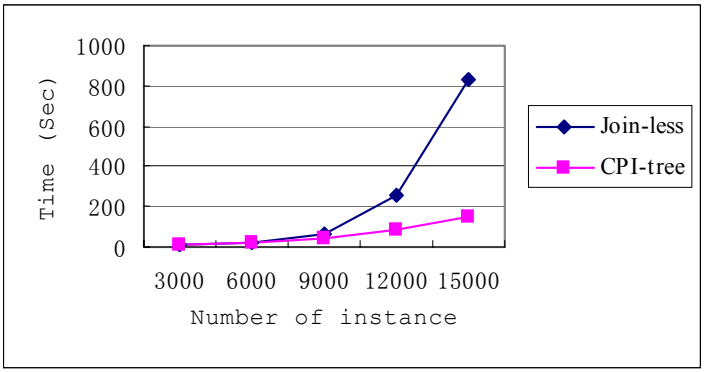


Figure 3.8 Scalability of CPI-tree algorithm with number of instances

3.6. Summary

In this Chapter, a new join-less co-location mining algorithm, which can rapidly generate spatial co-location table instances based on the *CPI-Tree* construction materialized neighbourhood relationship between spatial instances, was proposed. The algorithm is efficient since it does not require expensive spatial joins or instance join for identifying co-location table instances. The experimental results show the new method outperforms the join-less method in the case of sparse and dense datasets. As future work, the applications studying of co-location patterns mining is an important work. Treating with the redundant co-location rules and **maximal co-location patterns mining** will be significant works in the future work as well.

In next Chapter, an Order-Clique-Based Approach for Mining Maximal Co-locations will be discussed. The purpose of the studying is to more efficiently mining spatial co-location patterns.

Chapter 4

An Order-Clique-Based Approach for Mining Maximal Co-locations

Mining co-location patterns in spatial datasets have been studied popularly in spatial data mining research. Most of the previous studies adopt an *Apriori*-like approach to generate size-k prevalence co-locations after size-(k-1) prevalence co-locations. However, generating the prevalence co-locations and storing the excessive table instances is costly, especially when there are a large number of co-locations and table instances, and/or long patterns. A novel **order-clique-based approach for mining maximal co-locations**, which mines maximal co-locations without storing excessive table instances, is proposed. Characteristic and efficiency of the approach is achieved with three techniques: (1) the spatial neighbour relationships between instances and the size-2 prevalence co-locations are compressed into extended prefix-tree structures respectively, ***Neib-tree*** and ***P_2-tree***, which brings up a order-clique-based approach to mining candidate maximal ordered prevalence co-locations and ordered table instances, (2) all table instances are generated from the `Neib-tree`, the table instances **do not need be stored** after computing the *Pi* value of corresponding co-location, which dramatically reduces the executive time and space of mining maximal co-locations, and (3) some **strategies**, pruning the branches whose the number of child instances is less than a related value and scanning the `Neib-tree` in order, are used to stop some useless inspection in the process of inspecting table instances. The performance study shows that the method is efficient and scalable for mining both long and short co-location patterns, and is faster than the full-join method, the join-less method and the CPI-tree method.

4.1 Overview

Co-location patterns mining is a new branch studied in the spatial data mining field recently. A spatial co-location pattern represents a subset of spatial features whose instances are frequently located in a spatial neighbourhood. Spatial co-location patterns may yield important insights for many applications. For example, a mobile service provider may be interested in mobile service patterns frequently requested by geographical neighbouring users. The locations which are gotten together by people can be used for providing attractive location-sensitive advertisements, etc. Other application domains include Earth science, public health, biology, transportation, etc (Huang et al, 2004; Shekhar and Huang, 2001).

Co-location pattern discovery presents challenges due to the following reasons: First, it is difficult to find co-location patterns with traditional association rule mining algorithms since there is no concept of traditional “transaction” in most of spatial datasets (Koperski and Han, 1995; Wang et al, 2005). Second, the instances of a spatial feature distribute in spatial framework and share complex spatial neighbour relationships with other spatial instances (Chou, 1997; Cressie, 1991; Estivil-Castro and Lee, 2001; Estivil-Castro and Murray, 1998). So a large fraction of the computation time of mining co-location patterns is devoted to generating table instances of co-location patterns (Huang et al, 2004; Huang and Zhang, 2006).

In previous work on mining co-location patterns, Morimoto (2001) defined distance-based patterns called k-neighbouring class sets. In his work, the number of instances for each pattern is used as the prevalence measure, which does not possess an anti-monotone property by nature. However, Morimoto used a non-overlapping instance constraint to get the anti-monotone property for this measure. In contrast, Shekhar & Huang (2001) developed an event centric model, which does away with the non-overlapping instance constraint, and a new prevalence measure called the *participation index (Pi)* is defined. This measure possesses the desirable anti-monotone property. At the same time, Huang, Shekhar & Xiong (Huang et al, 2004) proposed a general mining approach: Join-based approach mining co-locations (called **join-based approach**), which established the basis of co-location mining. This approach is good on sparse spatial datasets. However, in dealing with dense datasets, it is inefficient due to the computation time of the join is growing with the growth in co-locations and table instances. Yoo and Shekhar proposed two improved algorithms (called **partial-join approach** and **join-less approach** respectively) to conquer the disadvantage of the full-join approach on efficiency in (Yoo and Shekhar, 2004) and (Yoo et al, 2005).

The partial-join approach is to build a set of *disjoint clique* in spatial instances to identify the intraX instances of co-location (belonging to a clique) and interX instances of co-location(belonging to between two cliques), and join the intraX instances and interX instances respectively to calculate the value of the *Pi*. This approach reduces the number of expensive join operations dramatically in finding table instances. However, the key of this algorithm is to find out cliques as big as possible, which could cut down the spatial neighbour relationships between two cliques. Besides building cliques is time-consuming, if the correct cliques could not be identified, and the number of cut neighbour relations would not be decreased, the partial-join algorithm of mining co-location pattern would be similar to the full-join algorithm.

The join-less approach puts the spatial neighbour relationships between instances into a compressed *star neighbourhood*. All the possible table instances for every co-

location pattern were generated by scanning the star neighbourhood, and by 3-time filtering operation. The join-less co-location mining algorithm is efficient since it uses an instance-lookup scheme instead of an expensive spatial or instance join operation for identifying co-location table instances. So the idea of the join-less is great. However, the star neighbourhood structure is not an idea structure for generating table instances, for the table instances generating from this structure have to be filtered. Therefore, the computation time of generating co-location table instances will increase with the growing length of co-location patterns.

To summarize above, in cases with a large number of prevalence co-locations and table instances, long co-locations, or quite low *min-prev* thresholds, the existing algorithms may suffer from the following two nontrivial costs:

(1). It is costly to handle a huge number of candidate co-locations. For example, if there are 10^3 spatial events, the algorithms will need to generate more than 10^5 size-2 candidates and test their occurrence prevalence. Moreover, to discover a prevalence co-location of size-100, such as $\{f_1,\ldots,f_{100}\}$, it must generate $2^{100}-2\approx10^{30}$ candidates in total. This is the inherent cost of generating the set of all prevalence co-locations, no matter what implementation technique is applied.

(2). It is wasteful to store excessive table instances of co-locations, which is especially true when the number of table instances is tremendous.

Can one substantially reduce the number of co-locations generated in prevalence co-location mining while preserving the complete information regarding the set of prevalence co-locations? Can one develop a method that utilizes some novel data structures and algorithms to avoid a huge number of table instances' store? This is the motivation on this Chapter.

In this chapter, maximal prevalence co-location mining substitutes for traditional prevalence co-location mining in order to solve the first question listed above. Some novel, compact data structures, $\mathtt{P_2\text{-}tree}$, $\mathtt{CP_m\text{-}tree}$, `Neib-tree` and `Ins-tree` is constructed, which extend prefix-tree structures storing crucial, quantitative information about size-2 prevalence co-locations, candidate maximal ordered prevalence co-locations, spatial neighbour relationships between instances and table instances. To ensure that the tree structures are compact and informative, the tree nodes are arranged in an ascending order (the spatial features in alphabetic order, and then the different instance of the same spatial feature in numerical order). Based on these tree structures, an *order-clique-based* approach mining candidate maximal ordered prevalence co-locations and generating table instances is developed, which use only the size-2 prevalence co-locations for generating candidate maximal co-location, while spatial neighbour

relationships between instances for generating table instances, perform iteratively mining maximal prevalence co-location patterns without excessive table instances' store.

A performance study has been conducted to compare the performance of the *order-clique-based* method with two representative co-location mining methods, the *full-join* and the *join-less*. The study shows that *order-clique-based* method is much faster than *full-join* and *join-less,* especially when the spatial dataset is dense (containing many table instances) and/or when the prevalence co-locations are long.

The remainder of the chapter is organized as follows. Section 4.2 introduces the concepts and lemmas of maximal ordered co-locations, and proposes algorithms of mining candidate maximal ordered prevalence co-locations based on ordered clique. Section 4.3 discusses table instances' inspection of candidate maximal co-locations based on ordered clique. Section 4.4 is the algorithm and analysis for mining maximal prevalence co-locations without redundant table instances' store. Section 4.5 presents the performance study. Section 4.6 gives the conclusions and discusses future work.

4.2 Maximal Ordered Prevalence Co-locations

In this section, the concept of maximal ordered prevalence co-locations is introduced, some lemmas are discussed, and algorithms for generating candidate maximal ordered prevalence co-locations are developed.

4.2.1 Definitions and Lemmas

Definition 4.1. GIven a co-location $c = \{f_1,..., f_k\}$ in a set of spatial events $F = \{f_1,..., f_n\}$, $k \in \{1,2,\cdots,n\}$, if $f_i \le f_j$ (in alphabetic order) holds for any $1 \le i \le j \le k$, the co-location *c* is called as an **ordered size-k co-location**. If *c* is a prevalence size-k co-location, it is called as an **ordered prevalence size-k co-location**.

Definition 4.2. Given a prevalence co-location $c = \{f_1,..., f_k\}$, $k \in \{1,2,\cdots,n\}$, $c \subset F$, if $c \cup f_i$ is a non-prevalent co-location for any $f_i \in F$ and $f_i \notin c$, the prevalent co-location *c* is called as a **maximal prevalence co-location**. If *c* is an ordered prevalence size-k co-location, it is called as a **maximal ordered prevalence co-location**.

Lemma 4.1. Let *R* be a set of all maximal ordered prevalence co-locations in finite spatial event set *F*, *c* is an ordered prevalence co-location, then $\exists w \in R$, $c \subseteq w$ holds.

Proof. (Antinomy) Suppose $\exists c$, which is a ordered prevalence co-location, for $\forall w \in R$, $c \not\subset w$ holds. From this supposition and definition 4.2, one can deduce that $\exists x_1 \in F$ and $x_1 \notin c$, $c \cup \{x_1\}$ is an ordered prevalence co-location. If the $c \cup \{x_1\}$ is a

maximal ordered prevalence co-location, then $c \subset (c \cup \{x_1\}) \in R$ holds. That infers antinomy. So, the $c \cup \{x_1\}$ is not a maximal ordered prevalence co-location. Therefore, $\exists x_2 \in F$ and $x_2 \notin c$, $c \cup \{x_1, x_2\}$ is an ordered prevalence co-location. That $c \cup \{x_1, x_2\}$ is not a maximal ordered prevalence co-location can be inferred using the same reason. In this way, there will be an infinite feature sequence $x_1, x_2, \cdots$. That is an antinomy due to the spatial feature set F is finite. That is to say, $\exists w \in R$, $c \subseteq w$ holds.

Lemma 4.1 points out that a set of maximal ordered prevalence co-locations contains any ordered prevalence co-location in finite spatial feature set *F*.

Definition 4.3. Given a set of ordered prevalence size-2 co-locations P_2, A set $\delta = \{\delta_{l_1}, \delta_{l_2} \ldots \delta_{l_m}\}$, where δ_{l_i} is the union of co-locations with the header feature l_i in P_2 and is sorted in ascending order, is called as the **size-2 co-location header relationship set**.

Example 4.1. Let an ordered prevalence size-2 co-locations set P_2 = {{*AB*}, {*AC*}, {*AD*}, {*BC*}, {*BD*}, {*BF*}, {*CD*}, {*CE*}, {*DE*}}, its size-2 **co-location header relationship set** δ **=**{ $\delta_A = \{ABCD\}, \delta_B = \{BCDF\}, \delta_C = \{CDE\}$ **,** $\delta_D = \{DE\}$ }.

Definition 4.4. Given the size-2 co-location header relationship set δ, a tree designed as below is called as a **prevalence size-2 co-location header relationship tree** (***P_2-tree***, for short).

1). It consists of one root labelled as "P_2", each element of size-2 co-location header relationship set is a sub-tree of this root.

2). A sub-tree consists of the root that is the header event and rest events as the children of the root.

3). Each node in the *HR sub-tree* consists of two fields: *event-name* and *node-link*, where *event-name* registers the event which this node represents, *node-link* links to the next node in the P_2-tree, carrying the same event-name, or null if there is none.

Example 4.2. Given a δ **=**{ $\delta_A = \{ABCD\}, \delta_B = \{BCDF\}, \delta_C = \{CDE\}, \delta_D = \{DE\}$ } obtained from an ordered prevalence size-2 co-location set P_2 = {{*AB*}, {*AC*}, {*AD*}, {*BC*}, {*BD*}, {*BF*}, {*CD*}, {*CE*}, {*DE*}}, the corresponding P_2-tree, which is constructed according to definition 4.4, is shown in Figure 4.1.

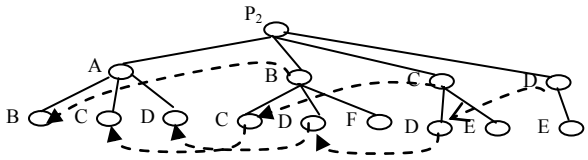


Figure 4.1 The P_2-tree of example 4.2

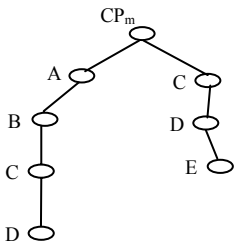


Figure 4.2 The CP_m-tree of P_2-tree in figure 4.1

Definition 4.5. Given an ordered size-k ($k \geq 2$) co-location c, c is called as a **candidate maximal ordered prevalence co-location** if c forms a **maximal clique** under the ordered prevalence size-2 co-location relationships.

Lemma 4.2. For any maximal ordered prevalence size-k ($k \geq 2$) co-location c, there must exist a candidate maximal ordered prevalence co-location, which is a super-set of c.

Proof. That any sub-set of a prevalence co-location is prevalent is proved in paper (Huang et al, 2004). So, any ordered size-2 co-location of a maximal prevalence size-k co-location is also prevalent. Therefore, under the ordered prevalence size-2 co-location relationships, the maximal prevalence size-k co-location forms a clique. Any clique can be extended to a maximal clique.

Definition 4.6. A tree, whose root is "CP_m" and each branch represents a candidate maximal ordered prevalence co-location, is called as a **candidate maximal ordered prevalence co-location tree (*CP_m-tree*,** for short).

Example 4.3. An example of the CP_m-tree is shown in Figure 4.2. One can see that each branch in the tree is a candidate maximal ordered prevalence co-location of the order prevalence size-2 co-locations {{*AB*}, {*AC*}, {*AD*}, {*BC*}, {*BD*}, {*BF*}, {*CD*}, {*CE*}, {*DE*}}.

Lemma 4.3. Let a P_2-tree, the process of producing a CP_m-tree from the P_2-tree is described as below.

1). Put the rightist child of the P_2-tree into H_c.

2). For child-node set ψ_c of each branch in H_c, for each node (denoted as H_c') linked out from the node-link of H_c, If some nodes (denoted as ψ_c') in ψ_c are brother-node of the node H_c', then ψ_c' is added as child-node of the H_c'.

3). Put the next child of the P_2-tree into H_c, go to 2), until the P_2-tree has not next child.

4). Delete all sub-trees which high is lower than 3 in the final CP_m-tree and redundant sub-trees (for example, there are candidate maximal prevalence co-locations {ABCD, ACD,…}, then "ACD" is redundant for it is contained by "ABCD".).

Proof. (a). If a candidate maximal prevalence co-location in CP_m-tree looks like 'ABC…', 'B' must be a child of the sub-tree A in P_2-tree, 'C' must be a child of the sub-tree B in P_2-tree, …et al. So, the step 1) and step 3) is proved.

(b). First, if ψ_x is a candidate maximal ordered prevalence co-location beginning feature 'x', the second feature 'y' in ψ_x should be a child of the node 'x' in P_2-tree. In other words, how many children the node 'x' has, how many probability of forming ψ_x has. Second, if ψ_x is an ordered co-location beginning feature 'x', and one of Tail (ψ_x) is not a child of 'x', ψ_x must be not an ordered prevalence co-location. Here explains the step 2) and 3).

(c). After the step 4), there are candidate maximal ordered prevalence co-locations which size is bigger than 2 in the CP_m-tree.

Example 4.4. The process of generating the CP_m-tree from a P_2-tree in Figure 4.1 is described as follows.

--Beginning the node '*D*' that is the rightist child of the root 'P_2' in Figure 4.1

--For first node (denoted '*D''* for conveniently) linked by '*D*', because child '*E*' of the '*D*' is the sibling of the node '*D''*', the node '*E*' is added as a child of the '*D''*'. For other two nodes linked by '*D*', nothing is done due to the condition is not satisfied.

--For branch '*CDE*', after executing the step 2) in Lemma 4.3, there is the result shown in Figure 4.3 (a). There is the result shown in Figure 4.3 (b) after finishing three children '*D*', '*C*' , and '*B*' of the 'P_2'.

--The final result is shown in Figure 4.2.

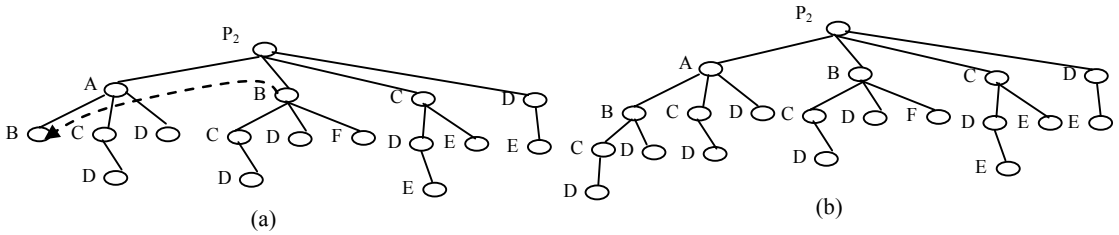


Figure 4.3. (a) is the result of after finishing two children 'D' and 'C' of the P_2-tree in Figure 4.1. (b) is the result of after finishing three children 'D' , 'C', and 'B' of the P_2-tree in Figure 4.1.

4.2.2 Algorithms

Based on the **definition 4.4**, an algorithm generating a ***P_2-tree*** from a set of ordered prevalence size-2 co-locations P_2 can be designed as below.

Algorithm 4.1 (Gen_P_2-tree)
Input: P_2: a set of ordered prevalence size-2 co-locations.
Output: P_2-tree: an ordered prevalence size-2 co-location header relationship tree.
Method:

```
1) Create a root "P2" for a new tree;
2) Let L=|P2|; /*L is the number of co-locations in P2
3) i=1; sub-tree="null";
4) While i≤ L Do; /*suppose P2(i,1) represents the first feature of the i-th co-location
   in P2, and P2(i,2) represents the second feature of the i-th co-location in P2.
5)  {if sub-tree≠P2(i,1) then
6)     {create a sub-tree "P2(i,1)" of the root "P2";
7)      the node-link of P2(i,1) is linked to the same-name nodes
   in P2-tree
8)     }
9)     Create a child-node P2(i,2) of the current sub-tree;
10)    i=i+1;
11)  }
12) Return the root 'P2'
```

Lemma 4.3 gives an idea for generating CP_m-tree from the P_2-tree. The following **algorithm 4.2** is designed based on **Lemma 4.3**.

Algorithm 4.2 (Gen_CP_m-tree)
Input: P_2-tree: an ordered prevalence size-2 co-locations header relationship tree.
Output: CP_m-tree: a candidate maximal ordered prevalence co-locations tree.
Method:

```
1) Let the sequence of all children of the root 'P2' in the P2-tree is {Hn,Hn-
   1,…,H1}; /*from the right child to the left child
2)  i=n;
3)  While i>1 Do
4)  { For each branch ψ of Hi
5)      For each node Hi' linked out from the node-link of Hi
6)        set(ψ)∩brother-set(Hi') is added as a new branch of the Hi';
7)    i=i-1;
8)  }
9)  Delete branches which level is lower than 3;
10) Delete redundant candidate co-locations from the CPm;
11)  Return the root 'CPm'
```

The correctness of this algorithm is guaranteed by **Lemma 4.3**.

4.3 Table Instances' Inspection of Candidate Maximal Co-locations

The basic idea of inspecting table instances of candidate maximal prevalence co-locations is similar to the idea of generating candidate maximal prevalence co-locations, since a table instance is a clique under the spatial neighbour relationships between instances. That is to say, to get candidate maximal prevalence co-locations is to compute **cliques** under the ordered prevalence size-2 co-location relationships, while to get table

instances is to compute **cliques** under spatial neighbour relationships between instances. So, there are some definitions and lemmas that are similar to Section 4.2.

4.3.1 Definitions and Lemmas

Definition 4.7. Given a subset of spatial instances $I = \{i_1,...,i_k\},\ k \in \{1,2,\cdots,m\}$, If $i_i \le i_j$ (the spatial events in alphabetic order, and then the different instance of the same spatial event in numerical order) holds for any $1 \le i \le j \le k$, then *I* is called as an **ordered size-k instance set**. If *I* is a table instance, it is called an **ordered size-k table instance**. If the event-name of i_i is not the same as the event-name of i_1 and $R(i_1, i_i)$ (represents i_1 and i_2 is neighbour) holds for any $1 < i \le k$, The *I* is called as **ordered neighbour relationship set of the instance** i_1. The set of ordered neighbour relationship sets of all instances of a spatial event *x* is denoted as δ_x**.**

Example 4.5. Let us take Figure 4.4 as an example. In this example, spatial event *A* has 4 instances, *B* has 5 instances, *C* is 3 instances, and *D* is 2 instances. Two instances are connected if they are neighbours in Figure 4.4. Therefore, $I = \{A.3, B.3, D.1\}$ is an ordered size-3 instance set, it also is an ordered size-3 table instance. The ordered neighbour relationship set of the instance A.3 is {A.3,B.3,C.1,C.3,D.1}. The set of ordered neighbour relationship sets of all instances of the event *A* is denoted as δ_A**={**{A.1,B.1,C.1}, {A.2,B.4,C.2}, {A.3,B.3,C.1,C.3,D.1}, {A.4,B.3}}.

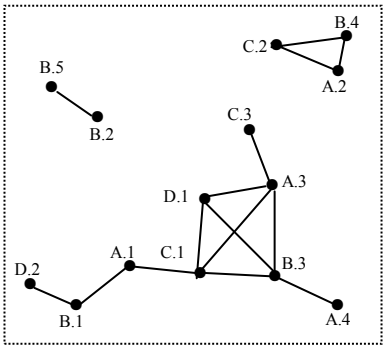


Figure 4.4 An example of spatial event instances

Definition 4.8. Given a set of spatial events $F = \{f_1,..., f_n\}$ and a set of ordered instance neighbour relationship of spatial events $\delta = \delta_{f_1} \cup \delta_{f_2} \cup ... \cup \delta_{f_n}$, $\delta_{f_i}(1 < i < n)$ is the set of ordered neighbour relationship sets of all instances of the event f_i, a tree designed as below is called as a **neighbour relationship tree** (`Neib-tree`, for short).

1). It consists of one root labelled as "*Neib*", a set of the spatial event sub-trees as the children of the root.

2). The spatial event f_i sub-tree consists of the root f_i and each subset of δ_{f_i} as a branch of the root. Each branch records ordered neighbour relationship set of corresponding instance.

Example 4.6. Figure 4.5 is the *Neib-tree* of the example in Figure 4.4. The event *'A'* sub-tree consists of the root *'A'* and branches *A.1, A.2, A.3,* and *A.4*. The branch *A.1* records the content of ordered neighbour relationship set of the instance *A.1*, i.e., there are $R(A.1, B.1)$ and $R(A.1, C.1)$.

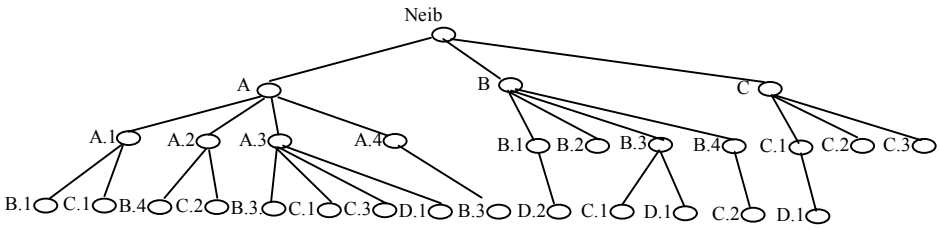


Figure 4.5 The Neib-tree of the example in figure 4.4

Definition 4.9. A tree, which root is "*Ins*", and each branch represents a candidate maximal ordered co-location's table instance, is called as a **table instances' inspection tree (***Ins-tree***,** for short).

Example 4.7. An example of the *Ins-tree*s is shown in Figure 4.6. It is *Ins-tree* of the candidate maximal ordered co-location {ABC} (suppose {ABC} is a candidate maximal ordered co-location of the example in Figure 4.4) of the *Neib-tree* in Figure 4.4. One can see that two table instances "A.2, B.4, C.2" and "A.3, B.3, C.1" are inspected in the *Ins-tree* of candidate maximal ordered co-location {ABC}.

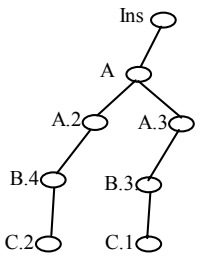


Figure 4.6 The Ins-tree of the candidate maximal prevalent co-location {ABC} of the Neib-tree in figure 4.5

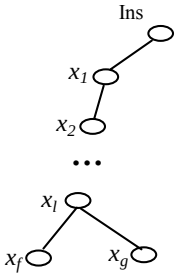


Figure 4.7 An explanation of Step 3) and 4) in Lemma 4.3

Lemma 4.4. Given a *Neib-tree*, the process of generating an *Ins-tree* from the *Neib-tree* is described as below.

1). Create a root "*Ins*" for inspecting a candidate maximal ordered co-location $c=\{f_1,f_2,\ldots,f_k\},\ k \in \{3,4,\cdots,n\}$. Copy sub-trees $f_1,f_2,\ldots,f_{k-1}$ in `Neib-tree` to the tree "*Ins*", at the same time, delete child-nodes that event-names of instances do not belong to $\{f_1,f_2,\ldots,f_k\}$, and delete branches of f_i $(1 \le i \le k)$, which child-nodes' number is less than (k-i).

2). For each child-node ψ_c of the branch L_c in the sub-tree f_1 of the tree "*Ins*", find out the same-name node ψ_c' (denoted it as ψ_c', for distinctively) of the sub-tree 'ψ_c' whose event-name is the same as the ψ_c.

3). For child-nodes set H_c of ψ_c', If some of H_c are brother-nodes of the ψ_c, the some of H_c can be moved to become child-nodes of the ψ_c from brother-nodes of the ψ_c and ψ_c becomes a extra-child of the L_c for the next loop. If the branch L_c still has child, go to 2).

4). Put the next branch to L_c, go to 2), until there is not next branch in the sub-tree f_1 of the tree "*Ins*".

5). Delete all sub-trees f_i $(2 \le i \le k)$ and all branches which high is lower than k.

Proof. (a). The instances of event f_i must be in sub-tree f_i, and if $i=\{i_{f_1},i_{f_2},\ldots,i_{f_k}\}$ (i_{f_i} is a instance of event f_i) is a ordered table instance of the $c=\{f_1,f_2,\ldots,f_k\}$, there exist $R(i_{f_i},i_{f_l})$ $(1 \le l \le k, l \ne i)$, that is to say that i_{f_i} at least has (k-i) children (they are $\{i_{f_{i+1}},i_{f_{i+2}},\ldots,i_{f_k}\}$). So, there is the step 1).

(b). Because any ordered table instance that starts an instance *x* must be by way of instances that they are child-node (neighbour) of the instance *x*. So, the step 2) is proved.

(c). If "$x_1,x_2,\ldots x_l\ x_f$" and "$x_1,x_2,\ldots x_l\ x_g$" is two ordered table instances (cliques) (shown in Figure 4.7), and $R(x_f,x_g)$ holds, then the "$x_1,x_2,\ldots x_l\ x_f\ x_g$" must be an ordered table instance (clique) (that principle is the same with the join principle of generating table instances in full-join method (Huang et al, 2004)). That is why here has the step 3) and 4).

(d). There is Step 5) just because one are just interesting in ordered table instances of the candidate maximal ordered co-location $c=\{f_1,f_2,\ldots,f_k\},\ k \in \{3,4,\cdots,n\}$.

Example 4.8. The process of generating the `Ins-tree` of a candidate maximal ordered co-location *c={ABC}* from the `Neib-tree` in Figure 4.5 is described as follows.

-- Create a root "*Ins*" for inspecting a candidate maximal ordered co-location *c={ABC}*. Copy sub-trees *A* and *B* in Figure 4.5 to the tree "*Ins*", at the same time, delete child-nodes that event-names of instances do not belong to {*A,B,C*}, and delete

branches of *A and B*, which child-nodes' number is less than 2 and 1 respectively. The result is shown in Figure 4.8 (a).

-- For child-node *B.1* of the branch *A.1* in the sub-tree *A* of the tree "*Ins*", there is not a same-name node in child-nodes of the sub-tree *B*. So, let us to handle the next child-node *C.1* of the branch *A.1,* there still nothing is done to the C.1 for the same reason. Then, go to the next loop (the branch *A.2* is considered).

-- In the branch *A.2,* first, for child-node *B.4* of the *A.2*, *C.2* is copied to become a child-node of the *B.4* due to *C.2 (the child of B.4')* is brother-node of the *B.4*. Second, for the next child-node *C.2* of the *A.2*, nothing is done for C.2′ is not in the tree "*Ins*". Third, after finishing all children of *A.2*, the branch *A.3* is considered…. after finishing the child-node *B.3* of the *A.3*, the result shown in Figure 4.8 (b) will be obtained.

--The final result is shown in Figure 4.6.

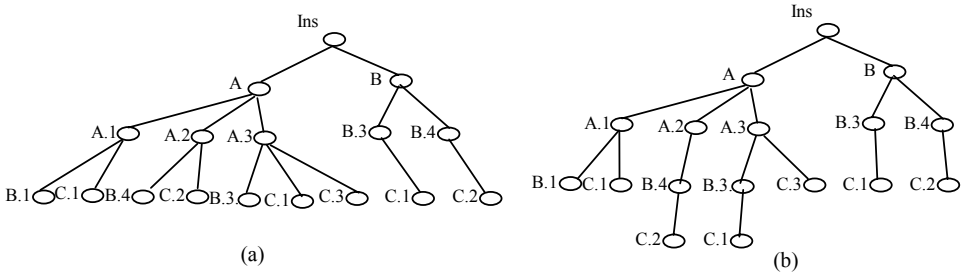


Figure 4.8 Two middle results in the process of generating Ins-tree of the co-location {ABC} from the Neib-tree in figure 4.5

4.3.2 Algorithms

According to the **definition 4.8**, an algorithm generating a ***Neib-tree*** from a set of ordered spatial neighbour relationships between spatial instances δ and a set of spatial events $F = \{f_1,..., f_n\}$ may be designed as below.

Algorithm 4.3 (Gen_Neib-tree)

Input: $F = \{f_1,..., f_n\}$: a set of spatial events.

$\delta = \{\delta_{f_1} = \{\delta_{f_1}^{l_1},...\delta_{f_1}^{l_{k_1}}\}, \delta_{f_2} = \{\delta_{f_2}^{l_1},...\delta_{f_2}^{l_{k_2}}\}...\delta_{f_n} = \{\delta_{f_m}^{l_1},...\delta_{f_m}^{l_{k_m}}\}\}$: a set of spatial ordered neighbour relationships between instances, where $\delta_{f_i}(1 \le i \le n)$ is the set of the set $\delta_{f_i}^{l_i}$ of ordered neighbour instances (they are "bigger" than the instance l_i) of instances l_i of feature f_i, whose order is sorted in ascending order.

Output: *Neib-tree*: an ordered instance neighbour relationship tree.

Method:

1) Create a root "Neib" for Neib-tree;
2) i=1;
3) **While** i<n **Do**;
4) **{** create a sub-tree f_i of the root "Neib";
5) Create a branch $\delta_{f_i}^{l_1}$ for sub-tree f_i;
6) **For** each $\delta_{f_i}^{l_i}\ (1 < i \le k_i)$ of δ_{f_i} in δ

7) create a child-node of the branch $\delta_{f_i}^{l_1}$;
8) i=i+1;
9) }
10) **Return** the root '*Neib*'

Lemma 4.4 gives an idea for generating ordered table instance of a candidate maximal ordered prevalence co-location C_m from corresponding *Neib-tree*, so an algorithm generating *Ins-tree* of the C_m is design as below.

Algorithm 4.4 (Gen_Instance)

Input: *Neib-tree:* an ordered instance neighbour relationship tree. C_m: a candidate maximal ordered prevalence co-location;

Output: *Ins-tree of the* C_m;

Method:

1) k=|C_m|; // N is the number of features in C_m
2) Create a root "Ins" for a Ins-tree;
3) **For** *i=1 to k-1 Do* /* $C_m(i)$ represents the i-th feature in C_m
4) {copy sub-tree $C_m(i)$ in *Neib-tree* to be a sub-tree of the root "*Ins*" ;
5) Delete instance-nodes whose feature do not belong to C_m;
6) Delete branches whose child-nodes' number is less than (k-i) }
7) **For** each branch L_c **in** *the sub-tree* $C_m(1)$
8) {**For** each child-node ψ_c **of** L_c
9) **If** find out the same-name-branch ψ_c' in the sub-tree feature-name (ψ_c) **then**
10) **If** some-child-nodes (ψ_c')= some-brother-nodes (ψ_c) **then**
11) {the some-brother-nodes (ψ_c) is moved to become child-nodes of the ψ_c;
12) the first child of the ψ_c becomes a child-node of L_c }
13) }
14) Delete branches whose level is lower than k+1;
15) **Return** the root '*Ins*'

The correctness of **Algorithm 4.4** is guaranteed by **Lemma 4.4**.

4.4 Algorithm and Analysis for Mining Maximal Ordered Prevalence Co-locations

Algorithms in Section 4.2 present candidate maximal ordered prevalence co-locations mining, while algorithm 4.3 and algorithm 4.4 in Section 4.3 generate table instance tree (*Ins-tree*) for a candidate maximal co-location. The algorithm and analysis for mining maximal ordered prevalence co-location are given in this section.

4.4.1 Algorithms

Based on algorithms presented in Section 4.2 and Section 4.3, the idea of mining maximal ordered prevalence co-locations is described in below.

(1). Computing spatial neighbour relationships between instances (the set of spatial neighbour relationships between instances is denoted as δ), and then the set of prevalent size-2 co-locations P_2 is generated.

(2). Based on the P_2, the *P_2-tree* is built by using **Algorithm 4.1,** and then the *CP_m-tree*, which contains the set of candidate maximal ordered prevalence co-locations CP_m, is generated by using **Algorithm 4.2.**

(3). Based on the δ, the `Neib-tree` is built by calling **Algorithm 4.3**

(4). Sorting CP_m in long descending order (in alphabetic order when their long are the same).

(5). For each candidate maximal ordered prevalence co-location C_m in CP_m, according to the value of C_m, the corresponding `Ins-tree` is generated by calling **Algorithm 4.4**, and then the value *Pi* of the C_m is computed. If C_m is not a prevalence co-location, the content of CP_m is changed (for a candidate maximal prevalence co-location set {ABCD, CDE,…}, if "ABCD" is not prevalent, it will be replaced by "ABC", "ABD", "ACD" and "BCD"). If C_m is prevalent, it is put to the set of maximal prevalence co-locations P_m.

(6). Return P_m

The idea above is transformed to the algorithm 4.5.

Algorithm 4.5 (`Gen_P`$_m$).

Input: A set of spatial dataset (including a set of spatial events $F = \{f_1,..., f_n\}$), a spatial neighbour distance threshold *d*, and a minimum prevalence threshold `Min-prev`.

Output: A set of maximal ordered prevalence co-locations with participation index is greater than `Min-prev`.

Method:

1) δ = *gen_neighbourhood(a spatial dataset);* P_m*=null;*

2) P_2*= gen_size-2_colocation(*δ*);*

3) *P_2-tree*=`Gen_P`$_2$`-tree(`*P_2*);* // calling **algorithm 4.1**

4) `CP`$_m$*-tree*=`Gen_CP`$_m$`-tree(`*P_2-tree*`)` // calling **algorithm 4.2**

5) `Neib-tree` =`Gen_Neib-tree` (δ, *F*); // calling **algorithm 4.3**

6) **For** each C_m in CP_m

7) **{** `Ins-tree` =`Gen_Instance`(`Neib-tree,`C_m); // calling **algorithm 4.4**

8) Compute `Pi(`C_m`)` from the `Ins-tree;`

9) **If** `Pi(`C_m`)` > `Min-prev` **then**

10) **{** P_m= $P_m \cup \{C_m\}$; delete the C_m from the CP_m **}**

11) **Else** the C_m in the CP_m is replaced by its sub-patterns;

12) **}**

13) **Return** P_m

4.4.2 Analysis

Time complexity: The time complexity of **algorithm 4.5** includes procedure *Gen_neighbourhood,* procedure *gen_size-2_colocation*, algorithm 4.1-3, and the loop 6) in algorithm 4.5. Suppose *m* is the total number of instances of all events and k is the number of intersections. In the worst case, the computational complexity of the procedure *Gen_neighbourhood* will be $O(m^2 \log_2 m)$, and the procedure *gen_size-2-_colocation* is $O(m \log_2 m + k)$ if the geometric approach proposed by Huang et al in (Huang et al, 2004) is used. For **algorithm 4.1 (**`Gen_P2-tree)`, if P_2 is sorted in alphabetic order, and *n* is the number of events, the complexity is $O(|P_2| + n^2 \log_2 n)$ (where $O(n^2 \log_2 n)$ is the cost of generating node-link of `P2-tree` in the worse case). For **algorithm 4.2** (`Gen_CPm-tree)`, the cost is $O(|P_2|)$ due to the number of branches in P_2 just be the number of co-locations in P_2. For **algorithm 4.3** (`Gen_Neib-tree`), if N_{ins} is the number of spatial neighbour relationships between instances, the cost is $O(N_{ins})$. For the loop of the step 6) in algorithm `Gen_Pm`, The cost consists of generating *`Ins-tree`* of a candidate maximal ordered prevalence co-location C_m by calling the **algorithm 4.4** and calculating the participation index of C_m.

Let us suppose that a set of spatial events $F = \{f_1, ..., f_n\}$, a set of spatial ordered neighbour relationships between instances $\delta = \{\delta_{f_1} = \{\delta_{f_1}^{l_1}, ...\delta_{f_1}^{l_{k_1}}\}, \delta_{f_2} = \{\delta_{f_2}^{l_1}, ...\delta_{f_2}^{l_{k_2}}\} ... \delta_{f_n} = \{\delta_{f_m}^{l_1}, ...\delta_{f_m}^{l_{k_m}}\}\}$, where $\delta_{f_i} (1 \le i \le n)$ is the set of the set $\delta_{f_i}^{l_i}$ of ordered neighbour instances of instances l_i of event f_i. If the longest candidate maximal ordered prevalence co-location is C_m in the CP_m, and the first event in the C_m is f_j, then, under the worse case, the cost of algorithm 4.4 is $O(|\delta_{f_j}| * |C_m|)$, while the cost of calculating the participation index of C_m is $O(|\delta_{f_j}| * |C_m|)$. The total cost of the loop of the step 6) in algorithm 4.5 `Gen_Pm` is less than $O(|\delta_{f_j}| * |C_m| * |CP_m|) \le O(N_{ins})$.

Summarizing above, the cost of **algorithm 4.5** `Gen_Pm` is T_{Gen_Pm} = $O(m^2 \log_2 m) + O(m \log_2 m + k) + O(|P_2| + n^2 \log_2 n) + O(|P_2|) + O(N_{ins}) + O(N_{ins})$

$\approx O(m^2 \log_2 m + n^2 \log_2 n + |P_2| + N_{ins}) \approx O(m^2 \log_2 m)$ due to $n << m$, $|P_2| < n^2$, and $N_{ins} < m^2$ hold.

It means that the efficiency of the algorithms depends on the number of spatial instances, the number of events, the number of size-2 co-locations, and the number of spatial neighbour relationships between instances, while the complexity mainly depends on the number of spatial instances. But by sorting spatial instances and co-locations, and using reasonable tree structures, which dramatically reduces the cost of algorithms. The real performance of algorithms is shown in Section 4.5.

Space complexity: The store space of the tree `Neib-tree` is the most costly in the algorithm, if it is always in the main memory, the space cost of the algorithm is $O(N_{ins}) \leq O(m^2)$. But a method which partial sub-trees of the `Neib-tree` are remained to reduce the need of the space can be adopted, because in one iterative of inspecting a candidate maximal ordered prevalence co-location C_m, the instances of events related to the C_m only need to be in the main memory.

4.5 Performance Study

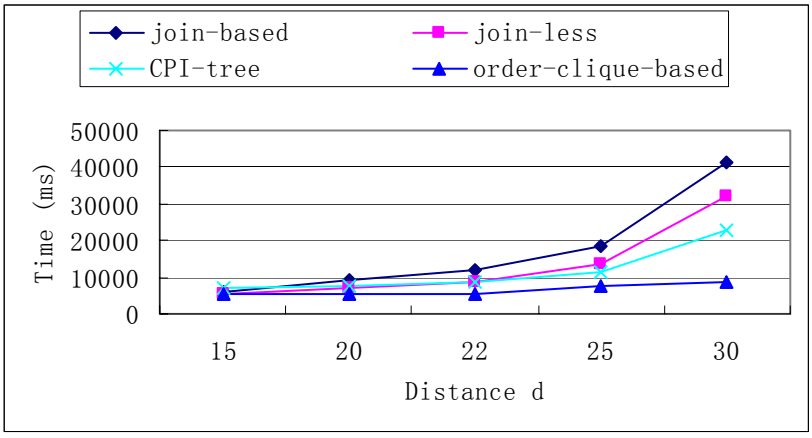
In this section, the performance of the **order-clique-based approach** presented in this chapter is evaluated with the **CPI-tree approach** discussed in the last chapter, the **join-based approach** and the **join-less approach** using both synthetic and real data sets. All the experiments were performed on a 3-GHz Pentium PC machine with 2G megabytes main memory, running on Microsoft Windows/XP. All programs are written in Java.

The experimental results are reported on two synthetic data sets. The first one is a sparse dataset with 26 spatial event types, when the neighbour distance threshold *d* and the prevalence threshold *min_prev* are set to 25 and 0.15, the total number of size 2 co-location patterns and the maximum size of co-location is 104 and 4, respectively, while the number of all instances in the dataset is 10k. The prevalent co-location patterns are short and not numerous in this dataset.

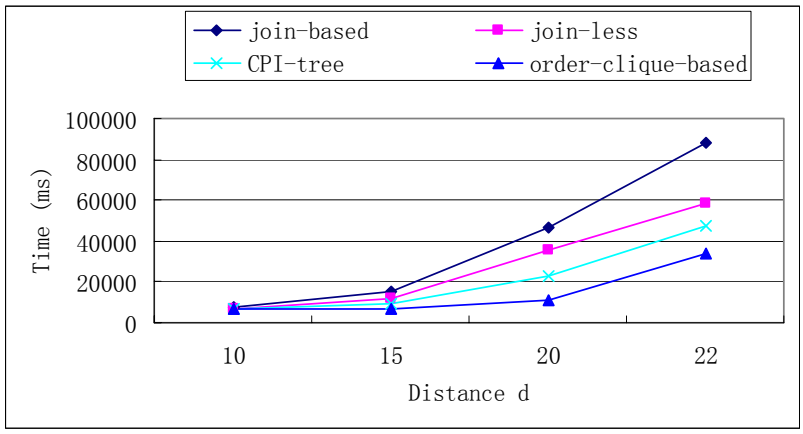
The second synthetic dataset used in the experiments is a dense dataset with 26 spatial events, the total number of size 2 co-location patterns and the maximum size of co-location is 232 and 8, respectively, when the threshold *d* and the *min_prev* are set to 25 and 0.15. There exist long prevalence co-locations as well as a large number of short prevalence co-locations in this dataset when the prevalence threshold *Min_prev* goes down.

To test the feasibility of the order-clique-based approach, a real dataset, the plant distributed data set of the “Three Parallel Rivers of Yunnan Protected Areas”, is used. It contains the number of plant species (feature types) is 16. The total number of plant instances is 3908. When *Min_prev* and distance *d* are set to 0.2 and 1700 respectively, the maximum size of co-location is 4 and the total number of size 2 co-location patterns is 39. There are a huge number of spatial neighbour relationships between instances due to the plants’ particularity of growing in groups.

1） Performance with the neighbour distance threshold *d* over sparse data set and dense data set: The runtime of the order-clique-based approach*, CPI-tree, join-based* and *Join-less* on the sparse synthetic data set, when the prevalence threshold *min_prev* is set to 0.15, as the neighbour distance threshold *d* increases from 15 to 30 is shown in Figure 4.9(a), while the results of the four methods on the abundant mixtures of short and long prevalence co-locations is shown in Figure 4.9(b) (the prevalence threshold *min_prev* is set as 0.15 in these experiments). Since the dataset is sparse, as the threshold *d* is low, the prevalent co-location patterns are short and the set of such patterns is not large, the advantages of order-clique-based over CPI-tree, join-based and join-less are not so impressive. However, as the threshold *d* goes up or the dataset becomes dense, the gap becomes wider. In fact, the number of co-location patterns of size 2 generated from the sparse data set is 75 when the threshold *d* is given 22 in the experiments. But the number has gone to 166 when the threshold *d* goes just to 22 in the dense data set.



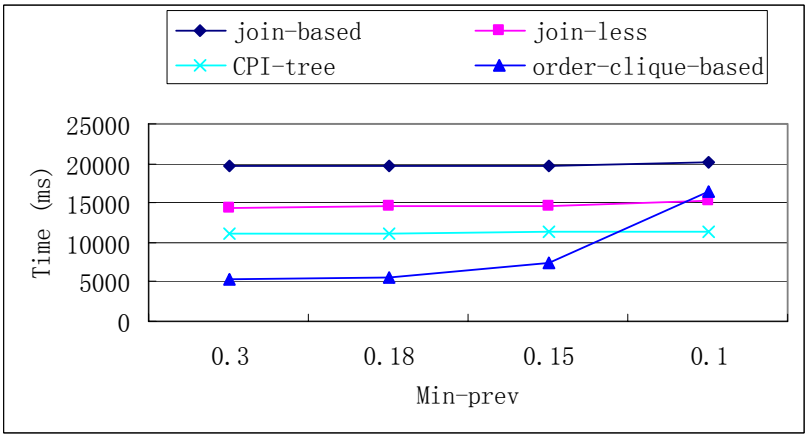
(a) Over a sparse dataset



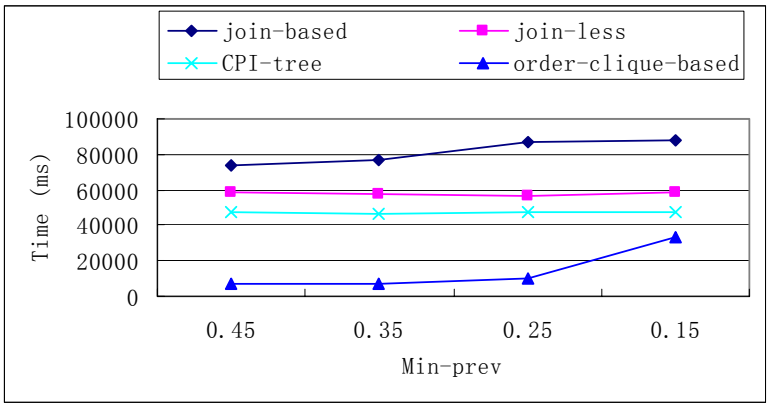
(b) Over a dense dataset

Figure 4.9 Scalability with *distance d* over a sparse dataset and a dense dataset

2）Performance with prevalence threshold *Min_prev* over sparse data set and dense data set: Figure 4.10(a) shows the experimental results with *Min_prev* over the sparse dataset, while the results over the dense dataset are shown in Figure 4.10(b). The neighbour distance threshold *d* is set to 25 in the experiments of Figure 4.10(a), while *d* is 22 in the experiments of Figure 4.10(b). In figures, it can be seen that the order-clique-based method has its advantage in most time. But it can also be found that sometimes it is not better than others, since the maximal co-location mining method might not be the best way in the case of shorter maximal co-locations.



(a) over a sparse data set



(b) over a dense data set

Figure 4.10 Scalability with *Min_prev* over a sparse dataset and a dense dataset

3）Performance with threshold *Distance d* over a real data set: The advantage of the order-clique-based method is dramatic in datasets with pretty long frequent co-location patterns, a large number of short frequent co-location patterns as well as a large number of table instances, which is challenging to the algorithms that mine the complete set of co-location patterns. The results over this real data set, a plants distributed dataset of the "Three Parallel Rivers of Yunnan Protected Areas", are shown in Figure 4.11. From the Figure 4.11, one can see that the new method is scalable even when there are many spatial neighbour relationships between instances due to the plants' particularity of growing in group. Without the excessive table instances' store and maximal ordered prevalence co-locations mining, the order-clique-based method and CPI-tree method are very efficient over real spatial datasets. In such datasets, the join-less method is not comparable to the performance of the order-clique-based method, and it is not even comparable to the performance of the join-based method.

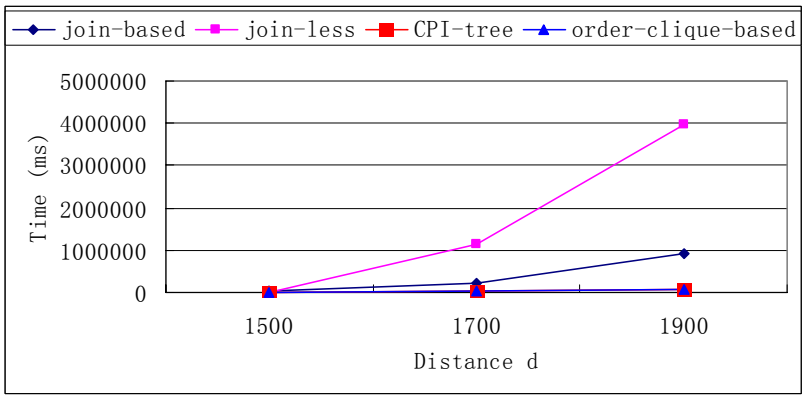


Figure 4.11 Scalability with *Distance d* over a plant distributed data set of the "Three Parallel Rivers of Yunnan Protected Areas"

4）Scalability of the order-clique-based method with number of instances: To test the scalability of the order-clique-based method against the number of instances, the dense dataset is used with *Min-Prev* is set to *0.3,* the neighbour distance threshold *d* is 20, and the number of instances ranges from 3K to 15K. The result is shown in Figure 4.12, which shows that the new method is the linear increase of runtime with the number of instances. At the same time, it shows that as the number of instances goes up, the join-less method is not comparable to the performance of the order-clique-based method.

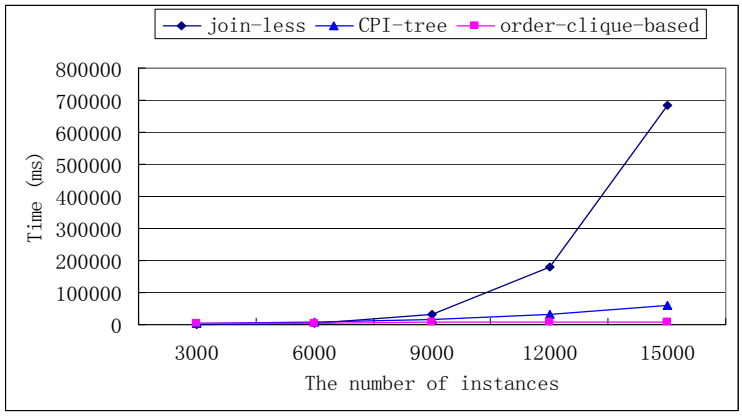


Figure 4.12 Scalability of the order-clique-based algorithm with number of instances

4.6 Summary

In this chapter, an order-clique-based method for mining maximal prevalence co-location, which can rapidly mining maximal ordered prevalence co-locations by adopting data structure P_2-tree, CP_m-tree, Neib-tree and Ins-tree to store related data information, and by sorting data in these tree, is proposed. The new approach is efficient since it does not require expensive table instances' joining and excessive table instances' storing operations to identify next level table instances. The experimental results show the order-clique-based method outperforms the CPI-tree, the full-join and the join-less methods in the case of sparse, dense datasets and real-world datasets. As future work, the applications studying of maximal co-location patterns mining is an important work. And **mining close co-location patterns** will be a significant work in the future works as well.

*In next Chapter, A reasonable new approach of AOI (attribute-oriented induction), attribute-oriented induction based on attributes' generalization sequences (AOI-ags), which expands the traditional AOI method, is proposed (*Wang et al, 2007(a))*.*

Chapter 5

AOI-ags Algorithms and Applications

The attribute-oriented induction (AOI, for short) method is one of the most important data mining methods. In this chapter, a reasonable approach of AOI (AOI-ags, attribute-oriented induction based on attributes' generalization sequence), which expands the traditional AOI method, is proposed. By introducing equivalence partition trees, an optimization algorithm of the AOI-ags is designed. Defining interestingness of attributes' generalization sequences, the selection problem of attributes' generalization sequences is solved. Extensive experimental results show that the AOI-ags are feasible. Particularly, by using the AOI-ags algorithm in a plant distributed dataset, some distributed rules for the species of plants in an area are found interesting.

5.1 Overview

The general idea of attribute-oriented induction (AOI) is to abstract each attribute of each record in a relation from a relatively low conceptual level to higher conceptual levels by using domain knowledge(Concept Hierarchy Trees), in order to discover rules among attributes from multilevel or higher level. The AOI approach to concept description was first proposed by Cai, Cercone, and Han in (Cai et al, 1991), a few years before the introduction of the data cube approach, and further extended by Han, Cai, and Cercone in (Han et al, 1993), Han and Fu in (Han and Fu, 1996), and Carter and Hamilton in (Carter and Hamilton, 1998). The data cube approach is essentially based on *materialized views* of the data, which typically have been pre-computed in a data warehouse. In general, it performs off-line aggregation before an OLAP or data mining query is submitted for processing. On the other hand, the attributed-oriented induction approach is basically a query-oriented, generalization-based, on-line data analysis technique.

There are two common ways to control a generalization process:

The first technique, called **attribute generalization threshold control**, either sets one generalization threshold for all of the attributes, or sets one threshold for each attribute. If the number of distinct values in an attribute is greater than the attribute threshold, further attribute generalization should be performed. If a user feels that the generalization reaches too high a level for a particular attribute, the threshold can be increased. Also, to further generalize a relation, the user can reduce the threshold of a particular attribute. But selecting an idea threshold for each attribute is not simple work.

The second technique, called **generalized relation threshold control**, sets a threshold for the generalized relation. If the number of (distinct) tuples in the generalized relation is greater than the threshold, further generalization should be performed. Otherwise, no further generalization should be performed. Such a threshold may also be preset in the data mining system by an expert or user, and should be adjustable. But the low-efficiency of AOI algorithms becomes the main problem in the relation threshold control, for it has to be considered that all combinations of attributes which satisfy the relation threshold in an attribute-oriented generalization process, and the results of generalization (or rank them for user) should be explained.

These two techniques can be applied in sequence: first apply the attribute threshold control technique to generalize each attribute, and then apply relation threshold control to further reduce the size of the generalized relation. No matter which generalization control technique is applied, the user should be allowed to adjust the generalization thresholds in order to obtain interesting concept descriptions. But it is still an intractable task to set the attribute thresholds in this combination of two techniques.

In this chapter, by introducing the concept of **attributes' generalization sequences**, the attribute threshold control technique and the relation threshold control technique are unified, and a new approach of AOI—AOI-ags (Attribute-Oriented Induction based on Attributes' Generalization Sequences), which expands the traditional AOI, is proposed. Some technologies, for example, partitions, equivalence partition trees, prune optimization strategies and interestingness, are used to improve the efficiency of the algorithm. It is shown that the AOI-ags algorithm has special advantages.

The rest of the chapter is organized as following. Section 5.2 formally defines the concept of the degree of relation generalization, and introduces the method of AOI based on attributes' generalization sequence (AOI-ags). In Section 5.3, by introducing equivalence partition trees, an optimization algorithm of AOI-ags is designed. Interestingness of attributes' generalization sequences is discussed in Sect. 5.4. Section 5.5 presents correctness, completeness and complexity of the algorithms. Performance and application results of algorithms are evaluated in Sect. 5.6. The last section is summary.

5.2 Attribute-Oriented Induction Based on Attributes' Generalization Sequences (AOI-ags)

AOI generalizes each attribute of each record in dataset from the lower conceptual levels up to the higher conceptual levels according to concept hierarchy trees and thresholds, to discover rules from multi-levels or higher levels. The concept hierarchy tree is an important part in AOI process. Some concept hierarchy trees can be obtained

from domain experts. But some concept hierarchy trees can be produced automatically from datasets by using hierarchical clustering approach (Lin and Chen, 2005).

For conveniently, some concepts about the concept hierarchy trees are defined as follows. The **Depth** of a certain node V in a tree is defined as the path length from root to V, the **Height** of V is the length of the longest path in the tree whose root is V. The **Height of the tree** is the Height of its root. The **Level** of V is its Depth more 1.

Example 5.1 Suppose the concept hierarchy tree of an attribute "elevation" in a relation table is shown in Figure 5.1, the Height of the concept hierarchy tree is 4. The Height of the node "[1500，2400]" is 0, its Depth is 4, its Level is 5. While the Height of node "[800，2400]" is 1, its Depth is 3, its Level is 4, and so on.

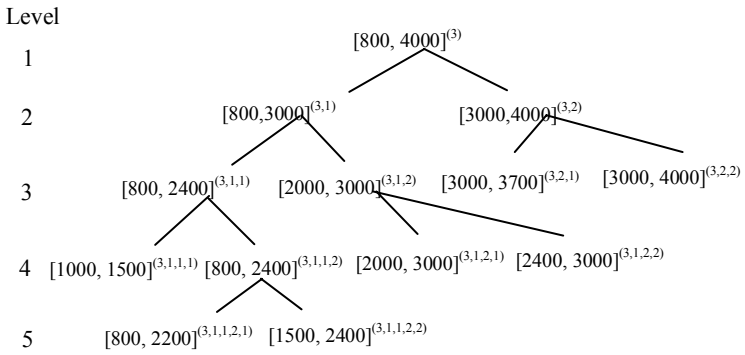


Figure 5.1 An example of a concept hierarchy tree

In traditional AOI algorithms, the generalization process is controlled by setting a threshold for each attribute. But in some applications, user does not want to consider each attribute for generalization threshold, so the degree of relation generalization is introduced.

Definition 5.1 Given a relation $r(r_1,...,r_n)$ and the generalization relation $r'(r_1',\cdots,r_{n'}')$, then the rate of reserved tuples is defined as $\overline{Z} = n'/n$, so the degree of relation generalization is defined as $Z = 1-\overline{Z} = 1-(n'/n)$.

Z is a measure for the degree of relation generalization. The higher the value of Z is, the greater the degree of generalization. The value Z meets $0 \le Z \le (n-1)/n$.

Z cannot confirm certain generalization result. That is to say, given a relation threshold control Z, some generalization relations that satisfy this threshold Z will be obtained. But analyzing the process of AOI, it can be found that generalization for each attribute is independent, that is to say, an attribute is generalized earlier or latter will not affect the final generalization result. Further to say, a generalization result is the same no matter that it is obtained by generalizing gradually or directly up to the k-th level, so at-

tributes' generalization sequences ("AGS" for short) is introduced in this chapter. One AGS confirms certain generalization relation.

Definition 5.2 Given a relation pattern $R(A_1,\cdots,A_m)$, attributes' concept hierarchy trees $h_1,\cdots,h_m$, the Heights of trees $l_1,\cdots,l_m$, sequence $A_1^{g_1}\cdots A_i^{g_i}\cdots A_m^{g_m}$ is called an AGS of AOI, where ($1\le g_i\le l_i+1$).

Property 5.1 The number of all AGS in a relation pattern is $\prod_{i=1}^{m}(l_i+1)$。

Proof. ∵ One sequence $g_1\cdots g_i\cdots g_m$ can only confirm one AGS $A_1^{g_1}\cdots A_i^{g_i}\cdots A_m^{g_m}$.

Meanwhile, ∵ $1\le g_i\le l_i+1 \quad \therefore |g_i|=l_i+1 \quad (1\le i\le m)$

∴ The number of attributes' generalization sequences is:

$$(l_1+1)\times\cdots\times(l_i+1)\times\cdots\times(l_m+1)=\prod_{i=1}^{m}(l_i+1) \qquad \square$$

Definition 5.3 Given the relation threshold control Z. If the generalization relation $r'(r'_1,\ldots r'_{n'})$ which are confirmed by the AGS $A_1^{g_1}\cdots A_i^{g_i}\cdots A_m^{g_m}$ ($1\le g_i\le l_i+1$) satisfies $1-(n'/n)\ge Z$, and if increasing any $g_i(1\le i\le m)$, it will not satisfy $1-(n'/n)\ge Z$, then $A_1^{g_1}\cdots A_m^{g_m}$ is called an AGS which satisfies the Z, and $r'(r'_1,\ldots r'_{n'})$ is called a generalization result under $A_1^{g_1}\cdots A_m^{g_m}$.

From Definition 5.3, it can be concluded that the AOI method of using attribute thresholds is a special case of using the relation thresholds. It means that the attribute thresholds and the relation thresholds are unified under the concept of AGS. The AOI based on AGS (for short, call it AOI-ags) is an extension to the traditional approach.

An ordinary AOI-ags algorithm is designed as follows.

Algorithm 5.1 The ordinary AOI-ags algorithm

Input:

- An un-generalized dataset (relation) r, which has m attributes {A_1,A_2….A_m}.
- Attributes' concept hierarchy trees $\{h_1,\cdots,h_m\}$ and the height of trees $\{l_1,\cdots,l_m\}$.
- The relation threshold control Z

Output: Generalization rules which meet the Z

Description:

1) Gen_seq(relation,1,m,L_1,S,*Gs*); //computing all attributes' generalization sequences *Gs* which meet the Z. the initial values are S="null", Gs=Φ.

2) Selecting a sequence from the set *Gs* of AGS and returning a generalization relation;

3) Producing generalization rules from the generalization relation.

```
Procedure Gen_seq(r,i,m,L_i,S,Gs); // S is an attributes' generalization sequence, Gs is a
collection of attributes' generalization sequences which meet the Z
(1)   For k=L_i+1 downto 1 Do
(2)     Begin If k<L_i+1 then
(3)              Gen_r ← generalize(r,i,k)
(4)           Else  Gen_r ← r
                Endif;
(5)           If i<m then
(6)               Gen_seq (Gen_r,i+1,m,L_i+1,S∪A^k_i,Gs)
(7)           Else
(8)             If |Gen-r|≤n(1-Z) then
(9)               Gs ← Gs∪ {S∪A^k_i}
                endif
              Endif
        End
```

When the number of attributes (m) is larger, in order to obtain all attributes' generalization sequences which meet the Z in algorithm, times $\prod_{i=1}^{m}(l_i+1)$ must be searched, and it will waste much time. So, how to efficiently compute all AGS which meet the Z is the chief problem in this algorithm. Further more, how to quickly calculate generalization relations that are related to AGS is another problem need to be solved. To solve these problems efficiently, an optimization AOI-ags algorithm is presented by introducing equivalence partition trees according to the property of AGS.

5.3 An Optimization AOI-ags Algorithm

5.3.1 AOI-ags and Partition

Let $r(r_1,\ldots,r_n)$ is a relation in relation pattern $R(A_1,\cdots,A_m)$, $X \subseteq R$, $\forall r_i, r_j \in r$, r_i and r_j are **equal** with respect to X if and only if $r_i[A_k] = r_j[A_k]$ for $\forall A_k \in X$, which is denoted as $r_i =_X r_j$. X partitions the rows of r into **equivalence classes**. The equivalence class of $r_i \in r$ with respect to $X \subseteq R$ can be denoted by $[r_i]_{=x}$. The quotient set $\pi_x = \{[r_i]_{=x} \mid r_i \in r\}$ of equivalence classes is an **equivalence partition** of r under X.

Given two partitions $\pi_x = \{\tau_1,\cdots,\tau_k\}$, $\pi_y = \{\tau'_1,\cdots,\tau'_{k'}\}$, $\pi_X \otimes \pi_Y = \{\tau_i \cap \tau'_j \mid \tau_i \in \pi_x, \tau'_j \in \pi_y\}$ is called **intersection partition** of π_X and π_Y, $\pi_X \otimes \pi_Y$ is a partition of r.

Property 5.2 $\pi_{X\cup Y} = \pi_X \otimes \pi_Y$

Proof. (1) $\pi_{X\cup Y}$ and $\pi_X \otimes \pi_Y$ are partitions

(2) $\forall \tau \in \pi_{x\cup y}$ and $\forall r_i, r_j \in \tau$, $r_i[X \cup Y] = r_j[X \cup Y]$ holds.

$\therefore r_i[X]=r_j[X]$ and $r_i[Y]=r_j[Y]$ hold.

Therefore $\exists \tau_1' \in \pi_x$ and $\tau_2' \in \pi_y$, $r_i, r_j \in \tau_1'$ and $r_i, r_j \in \tau_2'$ hold.

So, $\exists \tau'' \in \pi_x \otimes \pi_y$, $r_i, r_j \in \tau''$ holds.

(3) For $\forall \tau'' \in \pi_x \otimes \pi_y$ and $\forall r_i, r_j \in \tau''$,

$\exists \tau_1' \in \pi_x$ and $\tau_2' \in \pi_y$, $r_i, r_j \in \tau_1'$ and $r_i, r_j \in \tau_2'$ hold.

$\therefore r_i[X]=r_j[X]$ and $r_i[Y]=r_j[Y]$ $\therefore r_i[X \cup Y]=r_j[X \cup Y]$

$\therefore \exists \tau \in \pi_{x \cup y}$, $r_i, r_j \in \tau$ holds.

In fact, the equivalence class $\pi_R = \pi_{\{A_1, \cdots A_m\}}$ is the relation r. According to the property of $\pi_{X \cup Y} = \pi_X \otimes \pi_Y$, there is a one-one correspondence from the records of r to equivalence classes of $\pi_{\{A_1\}} \otimes \cdots \otimes \pi_{\{A_m\}}$. If $\pi_{A_i, j} (1 \le i \le m, 1 \le j \le l_i + 1)$ denotes equivalence partition which attribute A_i generalizes up to the j-th level along with the concept hierarchy tree, then the generalization relation and AGS $A_1^{g_1} \cdots A_m^{g_m}$ is corresponding one by one. This leads to a new partition-based approach of AOI-ags:

(1) Compute all $\pi_{A_i, g_i} (1 \le i \le m, 1 \le g_i \le l_i + 1)$.

(2) Obtain all AGS which meet the Z.

(3) Select a sequence $A_1^{g_1} \cdots A_m^{g_m}$, and then calculate generalization relation $r' = \pi_{A_1 g_1} \otimes \cdots \otimes \pi_{A_m g_m}$.

(4) Produce generalization rules from the generalization relation.

5.3.2 Search Space and Pruning Strategies

Definition 5.4 The Grid that is constituted by $\prod_{i=1}^{m} (l_i + 1)$ possible AGS and satisfies the following properties is called the **search space**.

(1) There are AGS that satisfy $g_1 + \cdots + g_m = k + 1$ in the k-th level.

(2) Each sequence is connected to any sequence $A_1^{g_1} \cdots A_i^{g_i - 1} \cdots A_m^{g_m}$ of the (k-1)-th level.

Example 5.2 Given two attributes A_1 and A_2, the Heights of the concept hierarchy trees are $l_1 = 2, l_2 = 3$, then the search space is showed in Figure 5.2.

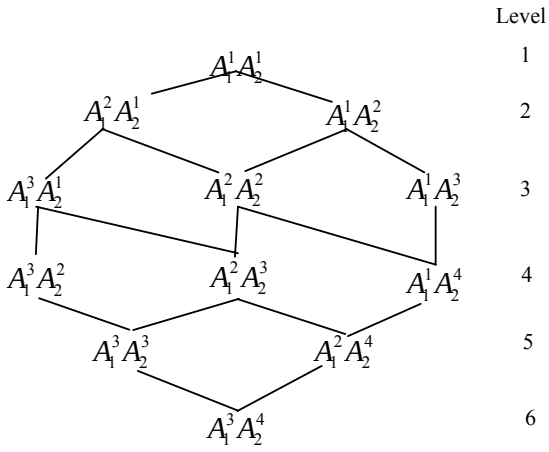


Figure 5.2 An example of the search space

The search space will increase rapidly with the increasing of m and l_i. By introducing the concept "refinement", some pruning is executed for reducing search space.

Definition 5.5 Given a relation and its two partitions $\pi_x = \{\tau_1, \cdots, \tau_k\}$, $\pi_y = \{\tau_1', \cdots, \tau_{k'}'\}$, if $\forall \tau_i \in \pi_X$, $\exists \tau_j^{'} \in \pi_Y$, $\tau_i \subseteq \tau_j^{'}$ holds, then π_X is called as a **refinement** of π_Y.

Obviously, $\pi_X \otimes \pi_Y$ is the refinement of π_X and π_Y, and $\pi_{A_i,j}$ refines $\pi_{A_i,k}$, $1 \le i \le m, 1 \le k < j \le l_i + 1$.

Property 5.3 If π_X refines π_Y, then $|\pi_X| \ge |\pi_Y|$ holds.

Proof. $\because \forall \tau_i \in \pi_X$, $\exists \tau_j^{'} \in \pi_Y$, $\tau_i \subseteq \tau_j^{'}$ holds,

And because π_X and π_Y are partitions.

$\therefore |\pi_X| \ge |\pi_Y|$ □

Definition 5.6 Given two sequences $A = A_1^{g_1} \cdots A_m^{g_m}$ and $A' = A_1^{g_1'} \cdots A_m^{g_m'}$, if $\forall A_i$, $g_i \ge g_i' (1 \le i \le m)$ holds, and then A is called a sub-sequence of A', denote as $A = sq(A')$, and A' is the parent-sequence of A, denoted as $A' = fq(A)$.

If $A = sq(A')$, then π_A refines $\pi_{A'}$. Therefore, what the pruning strategies can get are the following.

(1) If there exists a π_{A_i,g_i}, and $|\pi_{A_i,g_i}| > n(1-Z)$ holds, then any sequence which includes $A_i^{g_i}$ or its sub-sequence $A_i^{g_k}$ ($g_k \ge g_i$) cannot meet the Z.

(2) If there is a sequence $A_i^{g_i} \cdots A_j^{g_j}$, and $|\pi_{A_i,g_i} \cup \cdots \cup \pi_{A_j,g_j}| > n(1-Z)$ holds, then any sequence which includes $A_i^{g_i} \cdots A_j^{g_j}$ or its sub-sequence may not meet the Z.

(3) If a sequence $A = A_1^{g_1} \cdots A_m^{g_m}$ meets the Z, then all parent-sequences of A will not meet the Z, so it can be pruned.

The pruning strategies above, on the one hand, can form search lower bound in the search space, which reduces the search space, on the other hand, they can prune the attributes' generalization sequence which do not meet the threshold Z as early as possible, which avoids searching $\prod_{i=1}^{m}(l_i+1)$ times.

5.3.3 Equivalence Partition Trees and Calculating π_{A_i,g_i}

By introducing the concept of equivalence partition trees, π_{A_i,g_i} can be calculated efficiently. At first, each node (i.e., concept) in a concept hierarchy tree is assigned to a concept's code. As shown in Figure 5.1, the unary concept's codes (the numbers in bracket, "3" only represents the difference from other attributes) represent the first level in the concept hierarchy tree (i.e., the root of the tree). The binary codes represent the second level, etc.

Definition 5.7 The **equivalence partition tree** of the attribute A, each branch is a concrete value of A, which is concept's code with respect to the value in the concept hierarchy tree, each node in the tree is a value in concept's code.

Example 5.3 Take table 5.1 in Section 5.6.2 as an example, the equivalence partition tree of the attribute "elevation" is showed as Figure 5.3.

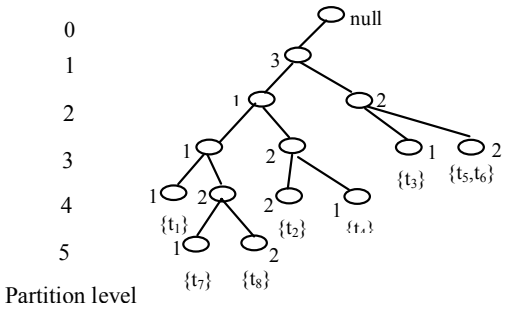


Figure 5.3 The equivalence partition tree of the attribute "elevation" in Table 5.1

An equivalence partition tree can be constructed by the following steps:

(1) Create the root of the tree, labelled with "null".

(2) For each value of the attribute, a branch is created.

(3) Note down the corresponding Row-id under the corresponding leaf node. The set of the Row-Id noted down is called the identity of the leaf-node.

Scanning a dataset once, the equivalence partition trees for each attribute can be constructed.

Definition 5.8 the **partition level** with respect to an equivalence partition tree is defined as follows. Define the root "null" as level 0, the following is level 1, …till the leaf node. The **identity of node** on an equivalence partition tree is the union of the identity of all leaf-nodes of the sub-trees with this node as their root.

It is not difficult to see that the partition level of the equivalence partition tree corresponds to the concept level of the concept hierarchy tree. The partition level with respect to equivalence partition tree of example 5.3 is shown in the left side of the Figure 5.3.

Property 5.4 the set of equivalence partition class with respect to attribute A at level *l* is the set of the identity of nodes of the *l*-th partition level in equivalence partition tree with respect to the attribute A (including leaf-nodes whose partition level is smaller than *l*).

Example 5.4 Considering the equivalence partition tree in Figure 5.3, the partition result in the third level is $\{\{t_1,t_7,t_8\}, \{t_2,t_4\}, \{t_3\}, \{t_5,t_6\}\}$, which is the same as the equivalence partition result after "elevation" is generalized to the third level.

5.3.4 Algorithms

Introducing partition and refinement, the AOI process can be speeded up efficiently. Introducing equivalence partition trees, equivalence partition results of attributes in any level can be quickly obtained. So an optimization algorithm of AOI-ags is designed as below:

Algorithm 5.2: An optimization algorithm of AOI-ags
Input:

- An un-generalized dataset r, which has m attributes $\{A_1,A_2…A_m\}$.
- Attribute's concept hierarchy tree $\{h_1,\cdots,h_m\}$ and their heights $\{l_1,\cdots,l_m\}$.
- The relation threshold control Z

Output: generalization rules which meet the Z

Description:
1) creation_partition_tree(r); //producing each attribute's equivalence partition tree
2) computing lower bound $L(A_i)$ which attribute A_i is generalized;
3) Gen($\pi_{A_i,l(A_i)}$,1,m,L(A_1),S,Gs); //the initial values are S="null", Gs=Φ. Obtain all AGS (Gs) which meet the Z
4) Selecting a generalization sequence $A_1^{g_1}\cdots A_m^{g_m}$ from Gs, computing generalization relation $r' = \pi_{A_1 g_1} \otimes \cdots \otimes \pi_{A_m g_m}$;
5) Producing generalization rules from the generalization relation.

Procedure creation_partition_tree(r)
Var item: Item; record: The set of item; T_1,…T_m: Tree;

```
Begin  create m root-nodes of m trees T1,…Tm, it is noted by
    "null" respectively;
  While  record:=get_next_record( r);
    For i:=1 to m do
      { item:=get_next_item (record);
        insert_tree (item,Ti); }
    End  {while}
  Return equivalence_partition _trees T1,…Tm.
End;
```

The *get_next_record* procedure reads a record and converts the attribute values to concepts. The *get_next_item* procedure gets an item (i.e., a concept) in the record, and converts the concept to concepts' code. And the *insert_tree (item, T_i)* procedure is performed as follows: Let the item be [E|item′], where E is the first element in the item. If T_i has a child N such that N.value=E.value, then it shares the node N; else create a new node N, its parent link to be linked to T_i. If item′ is nonempty, call insert_tree (item′, N) recursively. When the recursion procedure finishes, the Tuple-id is added to the identity of corresponding leaf-node.

```
Procedure Gen(r,i,m,L(Ai),S,Gs);
(1)   For k= L(Ai) downto 1 Do
(2)      Begin  If i=1 and k < L(Ai)   then
(3)                Gen_r ←π Ai,k
(4)             Else  If i=1 and k= L(Ai) then
(5)                      Gen_r ← r
(6)                   Else  Gen_r ← r ⊗ π Ai,k   Endif
                Endif;
(7)             if | Gen_r| > n(1-Z) then
(8)               exit for
                endif
(9)             If i<m then
(10)              Gen (Gen_r,i+1,m,L(Ai+1),S∪ A^k_i,Gs)
(11)            Else  If |Gen-r|≤ n(1-Z) then
(12)                    Gs ← Gs∪ {S∪ A^k_i};
(13)                    Exit for
                    Endif
                Endif
          End
```

In the optimization algorithm of AOI-ags, the recursive times are efficiently controlled by computing each attribute's lower limit $L(A_i)$, and consider whether AGS can be pruned or not in a recursive process according to *Pruning Strategies*. In fact, many recursive steps will be jumped over using pruning strategies in algorithm 5.2.

5.4 Interestingness of Attributes' Generalization Sequences

The method of attributed-oriented induction based on attributes' generalization sequence extends traditional AOI method, but which now induce a new problem: how to choose attributes' generalization sequence which have the same degree of generalization? So, the interestingness of attributes' generalization sequences is introduced.

Motivation example: For the plant "Magnolia sieboldii" in a plant distributed dataset, suppose the following rules have been obtained.

（1）Plant "Magnolia sieboldii" ⇒ 50% grows in the conifer forest and scrub whose elevation is from 2600 to 4100 meter of Lijiang, and 50% grows in the forest, scrub and meadow whose elevation is from 2400 to 3900 meter of Weixi.

（2）Plant "Magnolia sieboldii" ⇒ 90% grows in the conifer forest and scrub whose elevation is from 2600 to 4100 meter of Lijiang, and 10% grows in the forest, scrub and meadow whose elevation is from 2400 to 3900 meter of Weixi.

The rule (2) is more meaningful than the rule (1), because the growth characteristics of plant "Magnolia sieboldii" are more obvious in the rule (2).

Definition 5.8 In a generalization relation, the t-weight of the i-th generalization record t_i is defined as formula (1).

$$t_i = \frac{count\ (i)}{\sum_{j=1}^{n'} count\ (j)} \tag{1}$$

In formula (1), count (i) is the number of repeated records of the i-th generalization record in generalization relation, n' is the number of records in generalization relation.

Definition 5.9 Given $r'(r'_1,\dots r'_{n'})$ is a generalization relation under $A_1^{g_1}\cdots A_m^{g_m}$, ($1 \le g_i \le l_i + 1$), then interestingness $I_{g_1\cdots g_m}$ of $A_1^{g_1}\cdots A_m^{g_m}$ is defined as formula (2).

$$I_{g_1\cdots g_m} = \sqrt{\sum_{i=1}^{n'}(t_i - \frac{1}{n'})^2} \tag{2}$$

When the number of repeated records for each generalization record in a generalization relation $r'(r'_1,\dots r'_{n'})$ gets average value, $I_{g_1\cdots g_m}$ achieves the minimum 0. The farer t-weight of generalization records in a generalization relation is from average value, the larger the contribution to interestingness. The larger the value of $I_{g_1\cdots g_m}$, the more interesting the rule expressed by the attributes' generalization sequence $A_1^{g_1}\cdots A_m^{g_m}$.

Therefore, after obtaining sequences which meet the Z, computing their interestingness, and ranking the generalization sequences with the decline of interestingness, the generalization relation and rules can be produced.

5.5 Analysis

In this section, the algorithms are analyzed for completeness, correctness and computational complexity. Correctness means that the generalization rules meet the user specified threshold. Completeness implies that no AGS that satisfies the given threshold is missed.

5.5.1 Completeness and Correctness

Lemma 5.1 *Algorithm 5.1 is correct.*

Proof. The algorithm 5.1 uses very simple way to get generalization rules. It is obvious that algorithm 5.1 is correct if it can be proved that the recursive procedure Gen_seq is correct. That means the Gen_seq will return the AGS that satisfy the Z. It is guaranteed by step (8) in Gen_seq, because this step will check whether every sequence satisfies the Z or not.

Lemma 5.2 Algorithm 5.2 is correct.

Proof. The pruning strategy (1) guarantee the step 2) in algorithm 5.2 will return the low boundary of every attributes. If the return of step 3) is correct, then the Section 5.3.1 ensures the generalization relation computed by the step 4) is correct.

For step 3) (i.e., the recursive procedure Gen), the property of $\pi_{X\cup Y} = \pi_X \otimes \pi_Y$ and Section 5.3.2 ensure the correctness of the step (3)-(6) in Gen. The step (3)-(6) ensures the correctness of the step (11). And the step (11) guarantees every AGS satisfies the threshold Z.

Lemma 5.3 The algorithms are complete.

Proof. It will be proved that if a sequence satisfies the Z, it is found by the algorithms. In the recursive procedure Gen_seq of algorithm 5.1, the step (1) iterates all generalization levels of an attribute and the step (6) recursively perform Gen_seq. So the combine of step (1) and step (6) ensures the Gen_seq will check all possible candidate sequences.

For algorithm 5.2, the pruning strategy (1) guarantee the step 2) of algorithm 5.2 will return the low boundary of every attributes. In the recursive procedure Gen, The step (7) and (8) is because the pruning strategy (2) and the step (11) and (13) are just because

the pruning strategy (3). The combine of step (1) and step (10) guarantee the Gen will check all possible candidate sequences.

5.5.2 Computational Complexity

Suppose the number of records in the relation T is N, the number of attributes in T is m, and the height of attribute i-th concept hierarchy tree is l_i. In the worse case, the computational complexity of the algorithm 5.1 will be $O(N*\prod_{i=1}^{m}(l_i+1))$. And the algorithm 5.2 will be $O(\prod_{i=1}^{m}(l_i+1))$. Theoretically speaking, the computational complexity of the two algorithms seems not very distinctive. But many recursive steps will be jumped over using pruning strategies in algorithm 5.2, so the complexity will be much lower than algorithm 5.1. The real execution results will be shown in the Section 5.6.

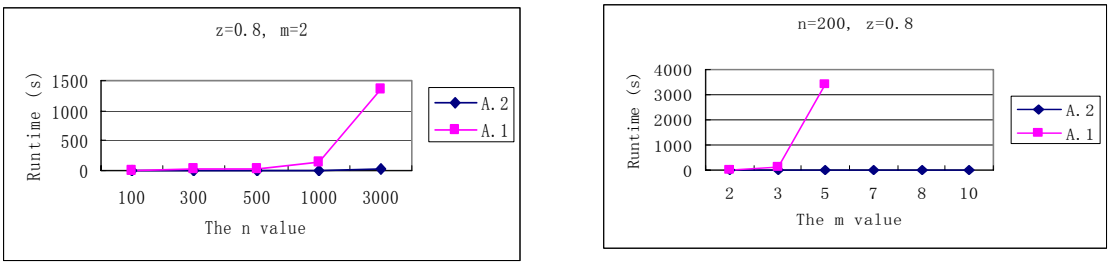
5.6 Performance Evaluation and Applications

The performance of the algorithms is evaluated by synthetic datasets and a real-world dataset (a plant distributed dataset of “The Three Parallel Rivers in Yunnan Protected Areas”). The experiments are performed on a Celeron computer with a 2.40 GHz CPU and 256 Mbytes memory running the Windows XP operating system.

5.6.1 Evaluation Using Synthetic Datasets

The experiments using synthetic data sets are aimed at answering the following questions. (1) How does the size of dataset affect the two algorithms? (2) How do the Algorithm 5.1 and the Algorithm 5.2 behave with the Z is changed?

A series of experiments are run with increasing number of spatial data points. The results are showed in Figure 5.4 (a). It can be seen that the algorithm 5.2 is almost linear and much faster than the algorithm 5.1.



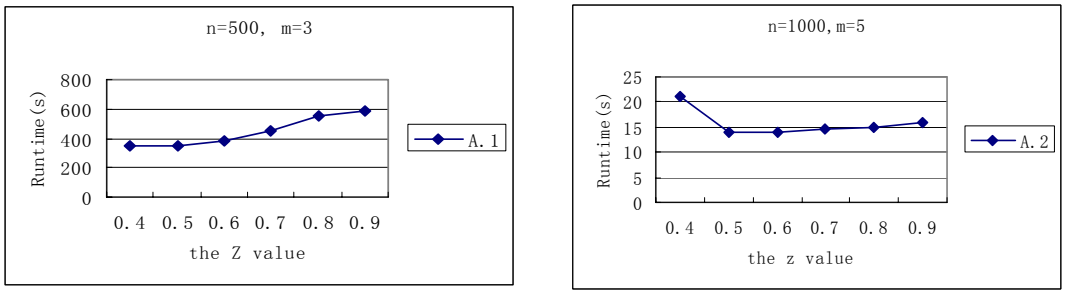
(a) (b)

Figure 5.4 Performance of algorithms using synthetic datasets

Fixed on the number of records, the number of attributes is an important parameter. The detailed comparative results are showed in Figure 5.4(b). It can be seen that the

performance of Algorithm 5.1 is very bad when m=5. The results indicate that the pruning strategies and equivalence partition trees used in Algorithm 5.2 are very efficient.

Now let us look at the characteristic of fast re-generalization of the two algorithms. The results are shown in Figures 5.5(a) (b). The 6 different settings of the thresholds Z are used in the experiments. It can be seen from the results that the Algorithm 5.2 possesses the characteristic of fast re-generalization.



(a) (b)

Figure 5.5 Characteristic of fast re-generalization for the two algorithms

5.6.2 Applications in a Real Dataset

A plant distributed dataset, which involves 29 plant species which are very valuable and rare in "The Three Parallel Rivers in Yunnan Areas" and 319 instances (tuples), is used in the experiments. Table 5.1 is some tuples of the dataset.

Table 5.1. Some tuples of a plant distributed dataset

Tuple-ID	Plant-name	Veg-name	Elevation /m	Location
t_1	Orchid	meadow	[1000, 1500]	Lijiang
t_2	Fig	scrub	[2400, 3000]	Weixi
t_3	Magnolia	scrub	[3000, 3700]	Lijiang
t_4	Calligonum	taiga	[2000, 3000]	Jianchuan
t_5	Magnolia	meadow	[3000, 4000]	Lanping
t_6	Agave	taiga	[3000, 4000]	Lanping
t_7	Yucca	forest	[1500, 2400]	Weixi
t_8	Waterlily	meadow	[800, 2200]	Jianchuan

The experiments using this dataset are aimed at checking the usefulness of the AOI-ags algorithms. Can they discover valuable patterns? Are the rules discovered by the algorithms interesting towards geographers and botanists?

34 AGS are obtained when the threshold Z is set to 0.8, and 57 plant distributed rules are discovered when one of the 34 AGS is chosen according to their interestingness. When the threshold Z is set to 0.85, the number of AGS is 28 and the number of rules is 19. When the Z is set to 0.9, the number of AGS and rules is 22 and 16 respectively.

Some rules discovered by the algorithms are really attractive to geographers and botanists. The following are some examples:

- "Tricholoma matsutake" ⇒ 40% grows in the forest and meadow whose elevation is from 3300 to 4100 meter of Lijiang.
- "Angiospermae" ⇒ 80% grows in the forest、scrub and meadow whose elevation is from 2400 to 3900 meter of Lijiang and Weixi.
- Lijiang ⇒ There are a plenty of plants species in severe danger such as "Tricholoma matsutake", "Angiospermae", "Gymnospermae", and so on.

5.7 Summary

Related approaches for mining the associations of attributes can be divided into the clustering-based approach, the association rule-based method and the approach of AOI. Clustering-based approach treats every attribute as a layer and considers clusters of point-data in each layer as candidates for mining associations (Estivill-Castro and Murray, 1998; Estivill-Castro and Lee, 2001). The complexity and the low-efficiency are the crucial problems of this method. The association rule-based approach is divided into the transaction-based method and distance-based method again. The transaction-based method computes the mining transaction (two-dimension table) by a reference-object centric model, so one can use the method which is similar to *Apriori* for mining the rules (Koperski and Han, 1995; Wang et al, 2005). The problem of this method is that a suitable reference-object is required to be specified. The distance-based method was proposed by Morimoto in (Morimoto, 2001), and Shekhar together with Huang in (Xiong et al, 2004; Huang et al, 2004; Yoo and Shekhar, 2004) did further research. Because of doing a plenty of join operations, executing efficiency is the key problem of this method. The approach of AOI is presented firstly by Cai, Cercone, and Han in (Cai et al, 1991). It is a simple and understandable method. But it is inconvenient because setting each attribute threshold is required.

The AOI-ags proposed in this chapter can obtain automatically rules under setting a threshold Z. Particularly, by using the AOI-ags algorithm in a plant distributed dataset, some distributed rules for the species of plants in "Three Parallel Rivers of Yunnan Protected Areas" are found interesting. The advantage of AOI method is that domain knowledge (concept hierarchy trees) is used in the process of data mining.

In next Chapter, A valuable Fuzzy Data Mining Prediction Technology*, the degree of fuzzy association based on the distribution of the variables for the prediction object and the concept of semantic proximity (SP) between two prediction objects, is discussed. Particularly, by using this technology in a system of predicting plant species in an ecological environment and a prediction system of shovel hoist cable service lifespan, the applied value of this method is verified (*Wang et al, 2007(b); Wang et al, 2007(c))*.*

Chapter 6

Fuzzy Data Mining Prediction Technologies and Applications

Based on a concept of fuzzy association degree, a data mining prediction method is proposed in this chapter. Inverse document frequency (IDF) weight function has been adopted in this investigation to measure the weights of conditional attributes in order to superpose the fuzzy association degrees. To implement the method, the “growing window” and the proximity computation pruning are introduced to reduce both I/O and CPU costs in computing the semantic proximity between time-series data. By applying the approach in a plant species and ecological environment dataset and a dataset for predicting shovel cable lifespan, one can see that the approach is reasonable and effective.

6.1 Overview

Databases are rich with hidden information that can be used for intelligent decision making. Data mining prediction is a form of data analysis that can be used to predict future data trends. There are a great number of methods in the mining prediction, such as statistical learning (Hastie et al, 2001), machine learning (Witten and Frank, 1999), decision tree (Alsabti et al, 1998), and fuzzy method (Liu and Song, 2001). Further more, many algorithms have been proposed that adapt association rule mining to the task of prediction. The CBA algorithm for associative classification was proposed by Liu, HSU, and Ma (1998). CPAR (Classification based on Predictive Association Rules) was proposed in Li, Han, and Pei (2001). Carter and Hamilton (1998) handle data generalization by the attribute-oriented generalization method. Wang (2000) proposes a fuzzy equivalence partition method to handle data generalization. A data reduction technique based on attribute-oriented generalization is presented in paper (Wang and Chen, 2005). Shi et al (2003) presents a novel data pre-processing technique called shrinking inspired by the Newton’s Universal Law of Gravitation in the real world, which optimizes the inner structure of data.

But no prediction method is superior over all others for all data types and domains. When a real application problem faced, any method mentioned above may be inapplicable to it. In this chapter, starting at a discussion of the two application problems, a reasonable and effective fuzzy data mining prediction approach is proposed, in order to resolve similar problems.

6.2 Preparing the Data for Prediction

This section describes issues regarding preparing the data for prediction by analyzing two practical application problems.

6.2.1 Preparing the Data for Predicting the Shovel Cable Lifespan

Electric cable shovels are the workhorses of ore-pit mining, each shovel handling mineral ore of many thousand tons per day. Ore is scooped into a shovel's bucket with digging energy supplied by motors pulling on cables of large diameter attached to the shovel's bucket. Cables (also called ropes) are expected to last for approximately 2000 hours of operation. However, current shovel cable lifespan can range from 400 to over 1800 hours over an entire shovel fleet.

Our goal is to predict the shovel cable lifespan. How the shovel is used for the most part determines cable's lifespan. From the shovel telemetry data, the information about shovel work status can be obtained. The sequence of activities in an ideal simple shovel duty cycle consists of digging, hoisting, swinging the dipper towards the truck, dumping and swinging back to the ground.

Several variables are believed to contribute to a cable's lifespan, and these variables are described as shovel dispatch data, shovel dig energy (energy expended within dig cycles), other shovel energy (energy expended outside dig cycles) and shovel-id (because it represents different shovel and different working site). There is intuitively a correlation between shovel dig energy and shovel cable lifespan. However the shovel dig energy alone cannot determine the exact shovel lifespan, nor can the shovel lifespan determine shovel dig energy. The same can be said for shovel dispatch data and other shovel energy. By analyzing the shovel motor performance data during shovel operation, the dig cycles of a shovel can be identified. So the dataset to predict the shovel cable lifespan is organized as Table 6.1.

Table 6.1. The dataset for predicting shovel cable lifespan

Tuple-No	**shovel dispatch data**	**shovel dig energy**	**other shovel energy**	**Shovel-ID**	**shovel lifespan**
r_1	127.3285,128.3771,137.5078, …	3437972,6070928,6087400, …	9102248,9471670,10623427, …	78	703.6
r_2	123.8424,127.9197,119.3427, …	3646337,3927044,4339156, …	10649652,8565822,8153072, …	78	1213.6
…	…	…	…	…	…
r_n	130.0234,128.3092,126.7473, …	4090329,4411364,3239778, …	9820630,9797128,10519956, …	84	639.2

In the dataset, there are two kinds of data. One is relatively static data that is stable and constrained to a finite number of values, for example, 'Shovel-ID' and "shovel life-

span". Others are time-series data. They change over time. If data were collected at a one-second time interval, then the quantity of data to be handled is huge. For example, the quantity of data of the shovel 78 in a lifespan (2005-04-24 to 2005-05-30) is 9*703.6*3600=22,796,640. So, obtaining a reduced representation of a time-series, comparing the similarity between two time-series, and partitioning the set of time-series, will be the works have to study in initial data exploration (IDE) for predicting the shovel cable lifespan.

6.2.2 Preparing the Data for Predicting Plant Species in an Ecological Environment

"Three Parallel Rivers of Yunnan Protected Areas", confirmed as the World Heritage on July 2, 2003 by UNESCO (United Nations Educational, Scientific and Cultural Organization), is the one of the most important researching areas for rocksy and botanist (Guo, 2004).

The ecological environments are believed to contribute to the plant species, distribution and diversity in Three-Parallel-River zone, which include climates (e.g., mean temperature, mean precipitation), elevation, topography, etc. So the data listed in Table 6.2 will be used in the chapter.

The ultimate goal of data mining in this plant species and ecological environment dataset is to discover the association pattern between plants and ecological environments in Three-Parallel-River Zone. Then it can be predicted that there might be some plant species in an ecological environment. Discovering correlations between the plant species and ecological environments will be very significant for retaining rare and endangered plants in Three-Parallel-River Zone.

Table 6.2. Plant species and ecological environment dataset

Tuple-ID	mean temperature (monthly 0.1^{o}C)	mean precipitation (monthly 0.1mm)	elevation (m)	topography	plant species
r_1	90, 100, 108, 130, …	0, 7, 21, 21, 307, …	[900,2000]	ascent	Camellia
r_2	80, 111, 130, 102, …	12, 13, 133, 55, …	[500,900]	ascent	Water-lily
r_3	99, 100, 144, 142, …	71, 205, 502, 330, …	[700,1100]	valley	Camellia
r_4	93, 115, 141, 165, …	0, 98, 171, 793, …	[200,700]	ascent	Camellia
r_5	77, 68, 116, 113, …	17, 228, 212, 453, …	[120,400]	valley	Water-lily
r_6	93, 105, 130, 145, …	36, 228, 679, 190, …	[200,800]	valley	Orchid
r_7	93, 103, 120, 151, …	40, 882, 46, 899, …	[600,1200]	basin	Orchid
r_8	67, 84, 81, 105, …	7, 62, 68, 184, 734, …	[1000,2000]	ascent	Water-lily

Prior to Table 6.2, data analysis is carried out as follows.

1). Climate data: there are two groups of time-series data for mean temperature and mean precipitation. These data change over time.

2). Elevation data: it is the interval values.

3). Topography data: it may have the following values: peak, valley, ascent, terrace, and basin…, all of them are used to describe the location of the plants growth.

There is hopefully a correlation between plants and ecological environments in Three-Parallel-River Zone. But a single ecological environment (e.g., elevation) cannot determine the exact plants growing; nor can the plants determine ecological environments. The same are climates and topography. The relationship between plants and the ecological environments in Three-Parallel-River Zone is a fuzzy association relationship.

The degree of fuzzy association represents the intensity of correlation between attributes in a data set. Therefore, in this chapter, the following problems need to be discussed:

1). How to explore the initial data? For example, the mean temperature of climate data is a time-series data.

2). How to evaluate the fuzzy association degree between the conditional attribute and the predicted attribute?

3). How to superimpose fuzzy association degrees?

6.3 Initial Data Exploration – IDE

In the light of the datasets above two applications, the key following issues are going to tackle in the study of IDE:

- Comparing the similarity between two time-series
- Approximately partitioning them

6.3.1 Comparing the Similarity of Two Time-Series

A time-series is a sequence of real numbers representing values at specific points in time. We start by defining time-series.

Definition 6.1 A time-series T of length n is an ordered set (t_1, t_2, … , t_n) with $t_i \in \Re$, $1 \le i \le n$. |T| is the length of T.

For comparing similarity of two time-series, the degree of proximity will be defined. The Euclidean distance, a popular similarity measure that has been extensively used in comparing time-series (Shi et al, 2003; Hastie et al, 2001; Witten and Frank, 1999), is adopted in this definition.

Definition 6.2 Given two time-series, T[1…n] and Q[1…n], the degree of proximity between T and Q (denoted PD (T, Q), $0 \le PD(T,Q) \le 1$) is defined as

$$PD(T,Q) = 1/\sqrt{\sum_{i=1}^{n}(t_i - q_i)^2 + 1} \quad (1)$$

Simple, yet important, a property held by the PD is described by the Lemma 6.1. This property will be useful in improving the performance of computing PD. Simply put the Lemma 6.1 states that if two time-series are in close proximity, then all their prefix subsequence of equal length are also in close proximity.

Lemma 6.1. If $PD(T[1...n], Q[1...n]) > \varepsilon$ for time-series T[1…n] and Q[1…n], $PD(T[1...k], Q[1...k]) > \varepsilon$ holds, for $1 \le k \le n$.

Proof. (By contradiction) If for a particular k, $1 \le k \le n$, $PD(T[1...k], Q[1...k]) \le \varepsilon$, however, $PD(T[1...n], Q[1...n]) \le PD(T[1...k], Q[1...k])$, and therefore $PD(T[1...n], Q[1...n]) \le \varepsilon$, a contradiction, since the $PD(T[1...n], Q[1...n]) > \varepsilon$ was assumed. □

Using a “growing window” to scan the time-series, the computation of PD(T, Q) can be done recursively by adding the remaining terms to the previously sums, thus the number of necessary computations are reduced. For example, if the values: PD(T[1…300], Q[1…300]) have been computed, then the values of PD(T[1…301], Q[1…301]) can be computed directly using Equation (2).

$$PD(T[1...301], Q[1...301]) = 1/\sqrt{\frac{1}{(PD(T[1...300], Q[1..300]))^2} + (T[301] - Q[301])^2} \quad (2)$$

This allows people to perform a “growing window” algorithm. For example, if we compute the arguments in (1) for a window of size m in T and Q, i.e., PD(T[1…m], Q[1…m]), we can compute the same arguments for the “growing” window PD(T[1…m+1], Q[1…m+1]) in O(1) time.

Using the “growing window” and Lemma 6.1, there is an efficient pruning strategy. When a value of PD from a growing window is less than ε, it can be considered as zero, and no more further computation.

In the above discussion, it is supposed that two time-series have the same length. However the situation is not always like this and the data one face are usually long and have noise. So, beside Lemma 6.1 and the “growing window”, here want to optimize the computation of PD by employing a method of scaling.

Let T [1..n] be a large time-series and m be an integer with 0<m<n. One wants to compress T from n to m points. By intuition, one can group sequential points of T and take their averages in order to form the smaller sequence. Figure 6.1 is an example for grouping a time-series. It presents a scaling of a 24-point time-series into 4 points.

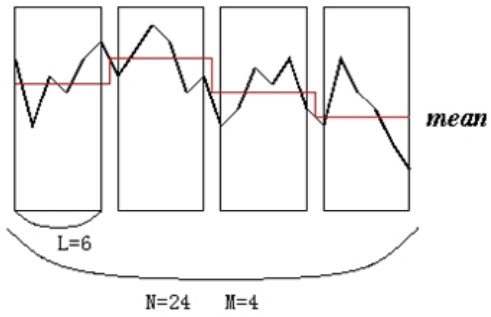


Figure 6.1 Scaling of a 24-point time-series into 4 points

Definition 6.3 Let T[1…n] be a time-series, m be an integer such that $1 \le m < n$, $A = \left\lceil \frac{n}{m} \right\rceil$, $B = \left\lfloor \frac{n}{m} \right\rfloor$ and $r = n \bmod m$. The scaling of T to size m, a time-series T_s[1…m] , is defined, where

$$T_s[i] = \begin{cases} \frac{1}{A} \sum_{j=(i-1)\cdot A+1}^{i\cdot A} T[j] & 1 \le i \le r \\ \frac{1}{B} \sum_{j=r+(i-1)\cdot B+1}^{r+i\cdot B} T[j] & r < i \le m \end{cases} \quad (3)$$

In the above definition, if n/m is not an integer, here take the first $n \bmod m$ points of the scaled sequence to be averages of A ($A = \left\lceil \frac{n}{m} \right\rceil$) points, and the rest to be averages of B ($B = \left\lfloor \frac{n}{m} \right\rfloor$) points of T. If $r = n \bmod m = 0$ holds, then the scaling will be consist of n/m points averages, as expected.

Taking averages has been used successfully as an approximation and dimensionality reduction technique in time-series (Witten and Frank, 1999; and Alsabti, 1998). This type of scaling that here used is robust to noise, which means that even small variances of the time sequence do not alter the scaled sequence. Moreover, it can be implemented in a computationally efficient way and transform the time-series to the same length. After all, the scaling procedure in this method is also similar to the human's nature understanding of "scaling".

6.3.2 Fuzzy Equivalence Partition for the Set of Time-Series

Once the value of PD between two time-series obtained, the remained issue is how to partition the set of time-series. Fuzzy equivalence partition method (Wang, 2000; Huo, 1989; and Zadeh, 1965) is used as follows.

Assume that $T_1, \cdots, T_N$ is a set of time-series. From Definition 6.2, there is a relationship $PD(T_i, T_i) = 1$ and $PD(T_i, T_j) = PD(T_j, T_i)$. Using the degree of proximity between two time-series, a similarity matrix $S = (s_{ij})_{N \times N}$, can be built up, where

$$s_{ij} = \begin{cases} 1 & i = j \\ PD(T_i, T_j) & i \neq j \end{cases} \qquad (4)$$

S can be multiplied by itself repeatedly, where $(s_{ij})^2 = MAX_k(MIN(s_{ik}, s_{kj}))$, until $S^{2k} = S^k . S^{2k}$ is called a fuzzy equivalence matrix *EM*. Based on the fuzzy equivalence matrix *EM*, the classifications of $T_1, \cdots, T_N$ will be obtained for user-specified level value λ.

The equivalence matrix *EM* can be computed in $O(N^3)$ time from a similarity matrix *S*. The computational method can be expressed as follows:

```
1 For i:=1 to N do
2   For j:=1 to N do
3     If s(j,i)>0 then
4       For k:=1 to N do
5          S(j,k):=max{S(j,k), min{S(j,i), S(i,k)}};
```

This algorithm is very efficient if the fuzzy similarity matrix *S* has many zero elements, due to step 3.

The above method can also be optimized through computing the fuzzy equivalence matrix *EM* in $O(M^2)$ time (*M* is the number of nonzero elements in the upper-triangle of the similarity matrix *S*.) (Wang, 2000).

6.3.3 An Example

Example 6.1 Let T be the set of the given time-series data T_1 – T_4 (see Table 6.3).

Table 6.3 A simple example of 4 time-series data

T	Time-series data
T_1	176.7289,137, 176.7289, 104, 120.9448, 137, 149.0745, 109.667,137, 137, 137, 170.2474, 108.6299, 163.7659, 181.9141, 160, 137, 85.5126, 160, 163.7659, 114.6793, 185.3709, 160, 180.1857, 137
T_2	177.5931, 104, 160, 144.7535, 107.7657, 140.0436, 160, 94.241, 160, 154.2597, 140.0004, 160, 160, 96.3151, 160, 158.5807, 133.0868,160
T_3	93.3336, 104, 160, 137, 156.8523, 100.6361, 160, 104, 137, 137, 133.0868, 158.1486,89.099, 160, 104, 120.988, 137, 137, 128.7658, 128.7658
T_4	181.9141, 104, 159.877, 160, 178.0252, 158.5807, 160, 137, 104, 137, 185.803,104, 137, 149.5066, 153.8276, 137, 160, 178.4573, 150.4572, 160, 186.6672, 176.2968, 137, 106.5127, 160, 160, 160,158.1486, 176.2968

Assuming $\varepsilon = 0.00001$ and m is 12 for this example. Then, from Definition 6.3, can get

T_{s1}={163.4859, 112.4724, 143.0373, 123.3335, 137, 139.4387, 172.8400, 148.5000, 122.7563, 139.2226, 172.6855, 158.5929};

T_{s2}={140.7966, 152.3768, 123.9047, 127.1205, 157.1299, 150.0002, 160, 96.3151, 160, 158.5807, 133.0868, 160};

T_{s3}={98.6668, 148.5, 128.7442, 132, 137, 145.6177, 124.5495, 112.494,137, 137, 128.7658, 128.7658};

T_{s4}={148.597, 165.5353, 133.6667, 142.2677, 146.7781, 148.5, 164.4572, 173.3336, 156.6484, 133.2564, 160, 167.2227}

From Definition 6.2, $s_{12}=PD(T_{s1},T_{s2})=1/[(163.4859-140.7966)^2+\ldots+(158.5929-160)^2+1]^{1/2}=0.0237$, $s_{13}=0.1282$, … .

Thus, $S=\begin{bmatrix} 1 & 0.0237 & 0.1282 & 0.0243 \\ 0.0237 & 1 & 0.0200 & 0.0120 \\ 0.1282 & 0.0200 & 1 & 0.0300 \\ 0.0243 & 0.0120 & 0.0300 & 1 \end{bmatrix}$

Applying S self-multiple repeatedly, can obtain $S^4=S^2=\begin{bmatrix} 1 & 0.0237 & 0.1282 & 0.0300 \\ 0.0237 & 1 & 0.0237 & 0.0237 \\ 0.1282 & 0.0237 & 1 & 0.0300 \\ 0.0300 & 0.0237 & 0.0300 & 1 \end{bmatrix}$

Then, S^4 is the fuzzy equivalence matrix of the S. When the level value λ is selected as 0.025, the level value matrix of the fuzzy equivalence matrix $S^4_{0.01}=\begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}$ can be obtained.

The classification of T_1 – T_4 is $T_{0.025}$={(T_1, T_3, T_4), (T_2)}. By the same method, $T_{0.1}$={(T_1, T_3), (T_2), (T_4)} could be obtained.

6.4. Mining Prediction

6.4.1. Degree of Fuzzy Association

Let $R=\{r_1, \ldots, r_p\}$ be a finite set of objects, $A=\{A_1, \ldots, A_n, B\}$ be a set of attributes over R. The attributes in A are classified into disjoin conditional attributes $C=\{A_1, \ldots, A_n\}$ and the predicted attribute $D=\{B\}$. The equivalent class L in the set of the equivalent classes for the conditional attributes A_k is denoted by A_{kL}, and for predicted attribute B, the B_m means the m^{th} equivalence class. The intersection of A_{kL} and B_m is denoted by $POS(A_{kL}, B_m)$. The number of objects in $POS(A_{kL}, B_m)$ is called the distribution of A_{kL} to B_m, written as $K(POS(A_{kL}, B_m))$.

The degree of fuzzy association between A_{kL} and B_m is determined as follows. Once the value of the conditional attribute A_k is A_{kL}, the value of the predicted attribute B must be B_m, then A_{kL} is in close association with B_m. In another condition, the conditional attribute A_k is A_{kL}, and the predicted attribute B is B_m or B_h, then the fuzzy association degree between A_{kL} and B_m can be decided depending on the proximity between B_m and B_h. The greater the proximity between B_m and B_h *is*, the higher the fuzzy association de-

gree between A_{kL} and B_m. If the B is B_m, B_h, or B_f when the A_k has the value A_{kL}, it is unlikely to have a relationship between A_{kL} and B_m.

Definition 6.4 Given a weight w_{jm} for measuring the proximity between the two values B_j and B_m of the predicted attribute B, the fuzzy association degree from A_{kj} to B_m is defined as ρ $(0 \le \rho \le 1)$

$$\begin{aligned}\rho(A_{kj}, B_m) = (K(POS(A_{kj}, B_1)) * w_{1m} \\ + K(POS(A_{kj}, B_2)) * w_{2m} \\ + \cdots + K(POS(A_{kj}, B_s)) * w_{sm}) / K(A_{kj})\end{aligned} \quad (5)$$

The weight w_{jm} , written as $w(B_j, B_m)$, represents the degree of proximity between B_j and B_m. $w(B_m, B_m)$=1 and $w(B_j, B_m)$=$w(B_m, B_j)$ are held. The weight w_{jm} can be obtained from domain expert (for example botanists), or by mining the co-location relation between plants (Huang et al, 2004; Yoo and Shekhar, 2004; and Xiong et al, 2004). The following two methods are adopted in the study.

1). Method for the plant data

Plants can be organized to a plant family tree, named as the conceptual hierarchy tree, with its leaf nodes describing an association relationship of plants. Figure 6.2 is an example of a plant family conceptual hierarchy tree

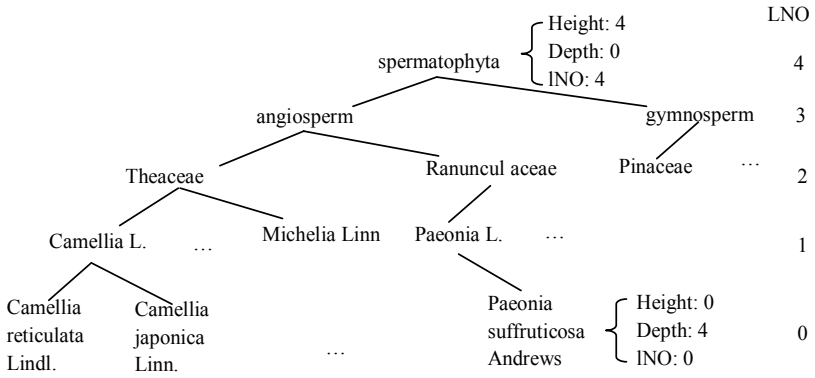


Figure 6.2 A concept hierarchy tree of plant species

Definition 6.5 The *Depth* of the node *v* in the concept hierarchy tree is the length of path form the root to this node *v*. The *height* of the node *v* is the length of the longest path in the subtree that the node *v* is the root. The *height* of the tree is defined as the height of the root. The *level number lNO* of the node *v* is the height of this tree minus the depth of *v*.

Example 6.2 Consider the concept hierarchy tree in Figure 6.2. Its height is 4. The height, depth and lNO of the nodes “Camellia japonica Linnn” and “spermatophyta” in

the concept hierarchy are shown in Figure 6.2 respectively. It can be seen that the lNO of nodes are given on the right side in Figure 6.2.

Definition 6.6 Given a conceptual hierarchy tree, f_1 and f_2 are two leaves of this tree, the value of weight $w(f_1, f_2)$ is defined as:

$$w(f_1, f_2) = \begin{cases} 1 & f_2 = f_1 \\ 1 - l(P(f_1, f_2))/H & f_2 \neq f_1 \end{cases} \quad (6)$$

Where, $P(f_1,f_2)$ returns the node of the common parent, $l(f)$ gets the level number of node f and H is the height of the tree.

Example 6.3 In the concept hierarchy shown in Figure 6.2,

$w(\text{Camellia reticulata Lindl}, \text{Camellia reticulata Lindl}) = 1$, and

$w(\text{Camellia reticulata Lindl}, \text{Paeonia suffruticosa Andrews}) = 1/4$

The weight w satisfies the following properties:

- For any plant f in a tree, $w(f,f)$=1 holds.
- For the parent of two plants f_1 and f_2 is the root in a tree, $w(f_1,f_2)=0$ holds.
- For any two plants f_1 and f_2 in a tree, $0 \le w(f_1, f_2) \le 1$.
- For any plants f_1, f_2, f_3 and f_4 in a tree, if $l(P(f_1,f_2))>l(P(f_3, f_4))$, then $w(f_1,f_2)<w(f_3,f_4)$.

2). Method for the lifespan data

For a value of the predicted attribute B, a center number (c, r) is used, c is the center of the sphere and r is the radius of the sphere, to describe it (e.g., a lifespan is 700 ± 12 hours), and the value of weight w_{jm} between two center numbers B_j and B_m can use the semantic proximity between B_j and B_m to measure.

Definition 6.7 The value of weight between two center numbers B_j and B_m can be defined as

$$w(B_j, B_m) = Length(B_j \cap B_m) / Length(B_j \cup B_m) \quad (7)$$

where Length(h) is the length of h.

6.4.2 Superposition of the Degrees of Fuzzy Association

The factors affecting the value of the predicted attribute B are not just the values of A_1, there are A_2, A_3…. How to superimpose them?

Definition 6.8 The superposition of $\rho_1, \rho_2, \ldots, \rho_k$, written as $\rho_1 \oplus \rho_2 \oplus \cdots \oplus \rho_k$, is defined as

$$\rho_1 \oplus \rho_2 \oplus \cdots \oplus \rho_k = [\sum_{i=1}^{k} \mu_i \rho_i^{\,2}]^{1/2} \qquad (8)$$

where μ_i is the weight on ρ_i.

Inverse document frequency (IDF) weight function is adapted to the relational domain by treating each tuple as document of attribute values. The motivation for this idea is clear from the following example. One expects the weight of attribute value 'a_1' to be less than that of 'd_3' since 'a_1' appears more frequently as a value for its attribute than 'd_3' does.

Definition 6.9 Let the frequency of attribute value '*a*' in the attribute A_i, denoted *freq (a, A_i)*, be the number of tuple (i.e., object) *r* in *R* such that *r[A_i]='a'*. The IDF value, *IDF(a, A_i)*, of an attribute value '*a*' with respect to the attribute A_i in the schema of *R* is computed as Equation (9), when *freq (a, A_i)>0*, (*K(R)* is the number of tuples in relation *R*)

$$\mu(a, A_i) = IDF(a, A_i) = \log \frac{K(R)}{freq(a, A_i)} \qquad (9)$$

For an attribute value '*a*' which's frequency in attribute A_i is 0, '*a*' is an erroneous version of some values in the reference tuple. Since one does not know the value to which it corresponds, the weight $\mu(a, A_i)$ is defined as the average weight of all values in this attribute of relation *R*.

6.4.3. An Example

Example 6.4 Suppose Table 6.4 is the result after preprocessing the data of Table 6.1 or 6.2. For Table 6.2, where A_1, A_2, A_3, A_4 represent the conditional attributes *mean temperature*, *mean precipitation*, *elevation*, and *topography* respectively, and attribute *B* is the *plant species*. The same lower case letter is used if $r_i[A_k]$, $r_j[A_k]$ belong to the same equivalence class (For the attribute *elevation* (A_3), it can be partitioned based on the concept of the semantic proximity between two interval values. In this study, the semantic proximity between two interval values f_1 and f_2 can be defined as $SP(f_1, f_2) = (size(f_1 \cap f_2) / size(f_1 \cup f_2))$, where $size(h)$ is the size of interval h).

There is a new conditional attributes data in Table 6.5 (it has been pre-processed). Let us predict the value of the predicted attribute B under these conditional attributes data. The distributing table from 'a_2', 'b_2', … to f_1, f_2 or f_3 (see Table 6.7) and corresponding values of *w* (see Table 6.6) are obtained, where for f_1 here have w_{11}= 1, w_{12}= 0, w_{13}=0, etc. In the same way, the $\mu(a, A_i)$ is computed, the result is Table 6.8.

Table 6.4 An example of data table for prediction

Tuple-ID	A_1	A_2	A_3	A_4	B
r_1	a_1	b_1	c_1	e_1	f_1
r_2	a_2	b_1	c_2	e_1	f_2
r_3	a_1	b_2	c_1	e_2	f_1
r_4	a_3	b_3	c_2	e_1	f_1
r_5	a_2	b_2	c_2	e_2	f_2
r_6	a_2	b_4	c_2	e_2	f_3
r_7	a_1	b_4	c_1	e_3	f_3
r_8	a_3	b_3	c_1	e_1	f_2

Table 6.5 A new conditional attributes' data that have been preprocessed

A new ecological environment data	A_1	A_2	A_3	A_4
r	a_2	b_2	c_1	e_1

Table 6.6 Weight *w* for the predicted attribute *B*

w	f_1 (Camellia)	f_2 (Water-lily)	f_3 (Orchid)
f_1	$1(w_{11})$	$0(w_{12})$	$0(w_{13})$
f_2	$0(w_{21})$	$1(w_{22})$	$0.6(w_{23})$
f_3	$0\ (w_{31})$	$0.6\ (w_{32})$	$1(w_{33})$

Table 6.7 The distributing table

K	f_1	f_2	f_3
a_2	0	2	1
b_2	1	1	0
c_1	2	1	1
e_1	2	2	0

Table 6.8 Weight μ for superposition

μ	a_2	b_2	c_1	e_1
	0.4260	0.6021	0.3010	0.3010

Thus here have

$\rho(a_2, f_1) = [K(Pos(a_2, f_1)) \times w_{11} + K(Pos(a_2, f_2)) \times w_{12} + K(Pos(a_2, f_3)) \times w_{13}] / 3$ =0*1+2*0+1*0= 0

$\rho(a_2, f_2) = [K(Pos(a_2, f_1)) \times w_{21} + K(Pos(a_2, f_2)) \times w_{22} + K(Pos(a_2, f_3)) \times w_{23}] / 3$ =(0*0+2*1+1* 0.6)/3=2.6/3=0.867

Using the same method, Table 6.9 is obtained. Superposing the data in row f_1, here have the result of association degree. It indicates the association degree of the conditional attributes data with 'a_2', 'b_2', 'c_1', and 'e_1' to predicted attributes data 'f_1', 'f_2', or 'f_3'. It means that the value of predicted attribute f_2 can be predicted in the conditional attributes data with 'a_2', 'b_2', 'c_1', and 'e_1'.

Table 6.9 Mining prediction of predicted attribute

ρ	a_2	b_2	c_1	e_1	The result of superposition
f_1	0	0.5	0.5	0.5	0.5487
f_2	0.867	0.5	0.4	0.5	0.7708
f_3	0.733	0.3	0.4	0.3	0.5986

6.5 Algorithms

The data mining prediction algorithm works in two procedures, the Initial_Data_Exploration procedure and the Data_Mining_Prediction procedure. The Initial_Data_Exploration procedure obtains the original data and partitions them into a relation table (see Table 6.4). The Data_Mining_Prediction procedure is invoked when a new conditional attributes data obtained. This procedure will predict the value of the predicted attribute for this new conditional attributes data.

6.5.1 IDE Algorithm

A simple yet efficient partition time-series data algorithm based on fuzzy equivalence partitioning according to the proximity degree of two time-series is designed in algorithm 1. It is a naive method that achieves its efficiency using “growing window”, scaling of time-series and pruning. As shown in Algorithm 1, it consists of three phases. In the first phase (line 1), it compresses and standardizes the input time-series. The case where the size of the time-series is larger than the memory buffer is considered (see the algorithm 1.1). In the second phase (line 2), it computes the proximity degree between time-series and forms the similarity matrix S. In the third phase (line 3), it performs the fuzzy equivalence partitioning process on the similarity matrix S and outputs the partition results. These phases are described in detail in Algorithm 1.1-1.3.

Algorithm 1 Initial_Data_ Exploration_Time-series;
Input: A set of time series data, $\{T_1, \ldots T_N\}$, with the length $\{n_1,\ldots n_N\}$ respectively, m: compress every T_i from n_i to m points, ε : 0.0001, λ : 0.01;
Output: fuzzy partition of its set;
Procedure
Begin
13) Scaling_Time-series;
14) Calculate_PD;
15) Partition;
End

Algorithm 1.1 Scaling_Time-series;
Input: A set of time-series data, $\{T_1, \ldots T_N\}$, with the length $\{n_1,\ldots n_N\}$ respectively, m: compress all T_i from n_i to m points;
Output: the set of time-series data each of which has m points, {Ts[1…N]};
Procedure
Begin
1) **For** i=1 **to** N **do**
2) $r = n[i] \bmod m$; $A = \left\lceil \frac{n[i]}{m} \right\rceil$; $B = \left\lfloor \frac{n[i]}{m} \right\rfloor$;
3) Read the first buffer from T_i;
4) **For** j=1 to r **do**
5) Ts[i, j]=0;
6) **For** k=(j-1)*A+1 **to** j*A **do**
7) **If** T[i, k] not in the buffer **then**
Read the next buffer from T_i
8) Ts[i, j]=Ts[i, j] + T[i, k];
9) Ts[i, j]=Ts[i, j] / A;
10) **For** j=r+1 **to** m **do**
11) Ts[i, j]=0;
12) **For** k=r+(j-1)*B+1 **to** r+j*B **do**
13) **If** T[i, k] not in the buffer **then**
Read the next buffer from T_i
14) Ts[i, j]=Ts[i, j] + T[i, k];
15) Ts[i, j]=Ts[i, j] / B;
End

Algorithm 1.2 Calculate_PD;
Input: the set of time-series data each of which has m points, {Ts[1…N]}, ε : 0.00001;
Output: the Similarity matrix $S = (s_{ij})_{N\times N}$;
Procedure
Begin
1) **For** i=1 **to** N-1 **do**
2) **For** j=i+1 to N **do**
3) Read the first 1/2 size of buffer from Tsi;
4) Read the first 1/2 size of buffer from Tsj;
5) $S[i, j]=1/\sqrt{(Ts[i,1]-Ts[j,1])^2+1}$;
6) K=2;
7) **While** S[i, j] > ε and $k \le m$ **do**
8) **If** Ts[i, k] not in the buffer **then**
Read the next buffer from Tsi and Tsj;
9) $S\{i, j\}=1/\sqrt{1/S[i, j]^2+(Ts[i,k]-Ts[j,k])^2}$;
10) K=k+1
Endwhile
11) **Endfor**
12) **Endfor**
End

Algorithm 1.3 Partition;
Input: the similarity matrix $S = (s_{ij})_{N\times N}$, λ :;
Output: fuzzy equivalence matrix EM; Equivalence partition classifications EPC;
Procedure
Begin
1) EM=S;
2) **For** i=1 **to** N **do**
3) **For** j:=1 **to** N **do**
4) **If** s(j,i)>0 **then**
5) **For** k:=1 **to** N **do**
6) S(j,k):=max{S(j,k), min{S(j,i), S(i,k)}};
7) **If** $S \neq EM$ **then**
EM=S goto step 2);
8) get EPC for λ
End

6.5.2 Mining Prediction Algorithm

The data_mining_prediction procedure performs mining prediction of predicted attribute. First, the relation table R is scanned by the Get_Attribute_Partition function to obtain the equivalence partition for every attribute in R. The detailed design of the Get_Attribute_Partition function is shown in Algorithm 2.1. Then all the weights w_{ij} between predicted attribute values i and j are computed. A result that looks like Table 6.6

will be obtained. Finally, the algorithm iteratively performs the following tasks for all new conditional attributes data. 1) Compute distributing table and weight μ according to the equivalence partition results in Step 1 of this algorithm; 2) compute the association degrees of this new conditional attributes data to every kind of predicted attribute value, and superimpose them; 3) output the results of mining prediction.

Algorithm 2 Data_Mining_Prediction;
Input: A relation table R in which all the data is preprocessed, con_attr_count: the number of condition attributes in table R;
Output: the prediction values of decision attributes for this new shovel data;
Procedure
Begin
1) Get_Attribute_Partition;
2) calculate_weight ω for the decision attribute;
3) Read 'a new ecological data' ED;
4) Transforming ED to useable form;
5) **While** not at end of input **do**
6) compute_distributing_table;
7) calculate_weight μ;
8) compute_assoiciation_degree ρ;
9) superpose_ associatione_degree;
10) output the result of prediction;
11) Read 'a new ecological data' ED;
12) Transformimg ED to useable form
Endwhile
End

Algorithm 2.1 Get_Attribute_Partition
Input: A relation table R in which all the data is preprocessed, con_attr_count: the number of condition attributes in table R;
Output: the set of equivalence classes of every attribute in R
Procedure
Begin
1) Read a record r from R;
2) **While** not at end of record **do**
3) **For** the value r[A] of each attribute A in r **do**
4) Replace_with_integer (r[A], I[A]); // to map the original value r[A] to a integer I[A] using a data structure such as a trie or a hash table//
5) The row r goes to IND(A)[I[A]]; // The I[A] equivalence class of attribute A contains the row r//
Endfor
6) Read a record r;
Endwhile
End

The Get_Attribute_Partition function computes equivalent classes of all attributes in relation R. A set of equivalent classes IND(A) with respect to an attribute A is computed from the column R[A] of relation R as follows. First, the values of the column are replaced with integers 1, 2, 3, … so that the equivalence relations do not change, i.e., same values are replaced with the same integers and different values with different integers. This can be done in linear time using a data structure such as a trie or a hash table to map the original values to integers. After this, the value r[A] is the number of the equivalence classes of IND(A) that contains the row r, and IND(A) is then easy to construct.

6.5.3 Analysis of Algorithm Complexity

Let us analyze the I/O and CPU cost of the Initial_Data_ Exploration. Suppose the number of records in R is N, the number of conditional attributes in R is M_{con} (in the application problems, it is 4), and the number of predicted attributes in R is M_{dec} (it is one in two applications mentioned in Section 6.2). If the number of points in every time-series has been scaled to m then the cost of Algorithm 1.1 is O(N*m), then, for a time-series

attribute, Algorithm 1.2 is $O(N^2*m)$, and Algorithm 1.3 is $O(N^3)$ (If the optimized method is used, the cost of algorithm 1.3 is $O(N^2)$). It is known that the cost of pre-processing other kind of attributes (e.g., interval values) will be much less than time-series data. So, the total cost of Algorithm 1 is at most $O(N^2*m* M_{con})$.

For the I/O and CPU cost of the Data_Mining_Prediction, In the Get_Attribute_Partition phase (Algorithm 2.1), each of the attributes for each record is sequentially scanned once, to get the set of equivalence classes for every attribute. The cost is $N*(M_{con} +M_{dec})$. In the calculate_weight ω phase, suppose the number of classes in the i-th predicted attribute is C_i. The number of weights need to be computed is C_i*C_i for the i-th attribute. But w_{ij} is equal to w_{ji}, and the values of the classes for the corresponding predicted attribute can be obtained from the previous phase, the cost of this phase is $\sum_{i=1}^{M_{dec}} \frac{C_i \times (C_i - 1)}{2}$. In the next phase, for a new ecological data, to compute the distributing table costs $O(M_{con} \times \sum_{i=1}^{M_{dec}} C_i)$, to compute weights μ costs $O(M_{con})$, to compute the degrees of association costs $O(M_{con} \times \sum_{i=1}^{M_{dec}} C_i)$ and to superimpose the degrees of association costs $O(\sum_{i=1}^{M_{dec}} C_i)$. The total cost is $O (M_{con} \times \sum_{i=1}^{M_{dec}} C_i)$.

Therefore, the cost of the Data_Mining Prediction algorithm is

$N* (M_{con} +M_{dec}) + \sum_{i=1}^{M_{dec}} \frac{C_i \times (C_i - 1)}{2} + M_{con} \times \sum_{i=1}^{M_{dec}} C_i$.

Since the number of predicted attributes is a small value (It is just one in two applications), and if C is the average value of C_i (i=1, …, M_{dec}), then the cost of the algorithm will become O (N*M) (M is the number of attributes in R).

6.6 Results of Experiments

In this section, the experimental results of the algorithms will be presented. More specifically, both the quality of results we came across and the performance of the algorithm will be illustrated. In the experiments the following data is used:

1). Rope history data;

2). Shovel dispatch data;

3). The raw telemetry data of shovels.

Seven different shovels collected from April 2005 to Feb. 2006 as used, and there shovel id's are 78, 79, 80, 82, 83, 84 and 85.

6.6.1 Estimating Error Rates

For estimating error rates, there are mainly three methods. One is called partition: training-and-testing. This method uses two independent data sets, e.g., training set (2/3), test set (1/3). It is usually used for data sets with large number of samples. Another method is called cross-validation. This method divides the data set into k sub-samples, and then, uses k-1 sub-samples as training data and one sub-sample as test data (it is called k-fold cross-validation). It is usually used for data sets with moderate size. The third is called bootstrapping (or leave-one-out cross-validation). It is used for small size data sets.

In this experiment, the bootstrapping method is chosen to estimating error rates. The first time, 13 lifespan have been chosen from shovel datasets. Then 34 lifespan were used to predict. A shovel cable lifespan is predicted according to the degree of association threshold. If the results of the algorithm are less than the threshold, this new shovel will be refused to predict. In the experiments, false positives (FP%) and false negatives (FN%) are used to evaluate the program's error rates.

Table 6.10 and Table 6.11 show the results of FP% and FN% with given the different thresholds of the fuzzy association degree for 13 and 34 lifespan. In the experiments, the equivalence partition level value thresholds λ_1 =0.00000114, λ_2=0.000002, λ_3=0.0045 is fixed for three time-series attributes respectively. It has been observed that to lower level the thresholds of fuzzy association degree will increases the percentage of false positive, and the result of 13 lifespan is better than 34 lifespan. Because in the experimental data, there are just 4 shovel sets of data and 7 classes of lifespan data in 13 lifespan experiments, and there are 7 shovels and 16 classes of lifespan in 34 lifespan experiments. The correlation of data in 34 lifespan is lower than in 13 lifespan, and the probability of prediction error in 34 lifespan is higher than in 13 lifespan. That means the data used in these experiments are not ideal. They are chosen just for they come from a real application.

Table 6.10 FP% and FN% for 13 lifespan

Thresholds of association degree / Error rates	0.6	0.5	0.4	0.3	0.2
FP%	22.2	27.2	27.2	23.1	23.1
FN%	75	100	100	null	null

Table 6.11 FP% and FN% for 34 lifespan

Thresholds of fuzzy Association degree / Error rates	0.6	0.5	0.4	0.3	0.2
FP%	37.5	57.8	56.5	55.6	55.6
FN%	57.8	75	75	null	null

6.6.2 Quality

Beside the error rate, other parameters have been used to measure the prediction performance, there are *Sensitivity=t_pos/pos, Specificity=t_neg/neg, Precision=t_pos/(t_pos+t_neg), and Accuracy=Sensitivity*(pos/(pos+neg))+Specificity*(neg/ (pos+neg)).*

Table 6.12 shows all results with using different measures for both 13 lifespan and 34 lifespan. In the experiments, the equivalence partition level value thresholds are fixed to λ_1 =0.00000114, λ_2=0.000002, λ_3=0.0045, and the degree of the fuzzy association threshold is 0.4.

Table 6.12 Results of the algorithm's quality

Quality / lifespan	Sensitivity	Specificity	Precision	Accuracy
13 lifespan	72.7	0	100	61.5
34 lifespan	76.9	25	90.9	40.7

The degree of fuzzy association threshold was set to 0.4, then all measures are observed with different equivalence partition level value thresholds for 13 lifespan (the results is shown in Table 6.13. The settings of equivalence partition level value thresholds are that λ^1 means λ_1=0.00000114, λ_2=0.000002, λ_3=0.0045, λ^2 means λ_1=0.000002, λ_2=0.0000025, λ_3=0.0045, and the λ^3 means λ_1=0.000003, λ_2=0.000003, λ_3=0.0045). It can be seen that the setting of equivalence partition level value thresholds is the most important step in this method. From experiments, the value of λ in row 1 of Table 6.13 produces the best outcome for the four measures given.

Table 6.13 The measure with different partition level value thresholds for 13 lifespan

Quality / partition thresholds	Sensitivity	Specificity	Precision	Accuracy
λ^1	8/11=72.7	0/2=0	1=100	8/13=61.5
λ^2	6/12=50	0/1=0	1=100	6/13=46.2
λ^3	5/13=38.5	0/0=null	1=100	5/13=38.5

6.6.3 Performance of the Algorithm

The algorithm was run on artificial datasets of size 13, 24, 33, …, 1000 lifespan. Figure 6.3 shows that the algorithm is increasing quickly with the increasing size of datasets, because the large time-series data need to be dealt with for just adding a tuple into the dataset.

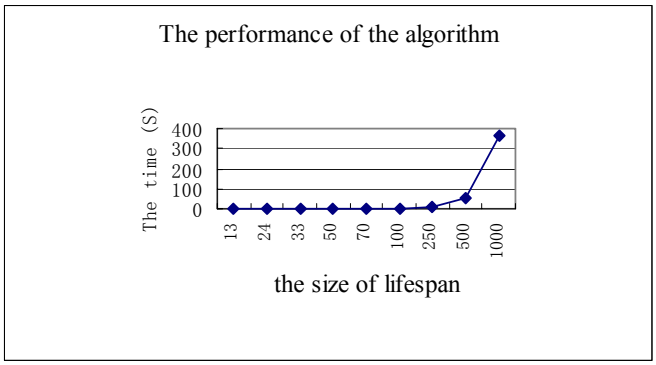


Figure 6.3 The performance of algorithm

6.7 Summary

This chapter makes contribution to using data mining techniques to resolve the predicting problem. It has been found that using the fuzzy association degree with superposition approach could achieve reasonable and effective results.

For the future work, it will be carried out formally to characterize the relative strengths and weaknesses of various prediction tests and to study the confidence of prediction results. Other interesting directions are mining a functional dependency relationship between conditional attributes, and mining crucial factors which affect plant growing, which could be advantageous to protect and retain rare and endangered plants.

In next Chapter, A new approach to deal with the object fusion problem, based on a special cell with a length that is equal to the error interval, is proposed. The key idea of this method is to find fusion sets by using cell-by-cell processing instead of object-by-object processing, thereby avoiding the computation of distance between two objects (Wang and Li, 2006).

Chapter 7

A Cell-Based Spatial Object Fusion Method

The object fusion problem occurred in geographic information system is also met in spatial data warehouses and is a very important problem in the spatial field. A cell-based spatial object fusion method in spatial data sets, which uses only locations of objects without distance between two objects involved, is described and its performance is measured in terms of recall and precision. This algorithm can work well when locations are imprecise and each spatial data set represents only some of the real-world entities. Results of extensive experimentation are presented and discussed.

7.1 Overview

With huge amounts of spatial data having been accumulated in the last two decades by government agencies and other organizations for various purposes such as land information management, asset and facility management, resource management and environment management, it is a pressing task to integrate information from heterogeneous information sources. When integrating data from heterogeneous information sources, one is faced with the task of fusing distinct objects that represent the same real-world entity. This is known as the **object-fusion problem**.

In the various researches on object fusion, some have considered that objects have identifiers (e.g., keys) (Papakonstantinou et al, 1996; Smal et al, 2004), while in (Beeri et al, 2004) and in (Minami, 2000) studied this problem without global identifiers. The lack of global identifiers makes the object-fusion problem much more difficult. In addition, in Bruns's paper (Bruns and Egenhofer, 1996), topological similarity is used to find corresponding objects, while ontology is used for that purpose in (Fonseca and Egenhofer, 1999; Fonseca et al, 2002; and Uitermark et al, 1999). Finally, the problem of how to fuse objects, rather than how to find fusion sets, was studied in (Papakonstantinou et al, 1996).

A spatial database or data warehouse stores spatial objects, or objects for short. Each object represents a single, real-world, spatial entity. An object has associated spatial and non-spatial attributes. Spatial attributes describe the location, height, shape and topology of the entity, while non-spatial attributes are usually place-name, temperature, humidity, etc. Object fusion is much harder without global identifiers. When fusing objects, spatial and non-spatial attributes should be used in lieu of global identifiers. Since

location is the only property that is always available for spatial objects, the location-based fusion problems are investigated, assuming that each dataset has, at most, one object per real-world entity, and locations are given as points. Thus, the fusion problem in this research is one-to-one.

Using only object's location, many efficient algorithms have been developed, such as the one-sided nearest-neighbour join (Minami, 2000), the mutually-nearest method (Beeri et al, 2004), the probabilistic method and the normalized-weights method (Beeri et al, 2004). The mutually-nearest method is an improvement of the one-sided nearest-neighbour join, and the probabilistic method and the normalized-weights method are based on a probabilistic model which are shown in (Beeri et al, 2004) achieve the best results under all circumstances. Although these methods are very fresh and novel, they need to compute the distance between two objects. This unfortunately is not a simple task, because the locations of objects are spatial attributes. As an alternative, a cell-based algorithm for finding corresponding objects that should be fused is presented in the chapter. In the approach, a special cell is defined. The cells possess some peculiar properties used to help finding fusion objects. The results of extensive tests that illustrate the validity and efficiency of this algorithm are also presented.

The main contribution of the work is finding corresponding objects effectively without the distance between two objects. The rest of this chapter is organized as follows. In Section 7.2, the problem is defined formally and how to measure the quality of the result of a fusion algorithm is described. Section 7.3 describes the fusion algorithm proposed. The testes and their results are discussed in Section 7.4. The summary is written in Section 7.5.

7.2 Basic Definitions and Measurements

In general, a fusion algorithm may process more than two datasets, generating fusion sets with, at most, one object from each dataset. In the research, the case of two datasets and investigate the problem of finding the correct fusion sets are considered, under the following assumptions. First, in each dataset, distinct objects represent distinct real-world entities. This assumption is realistic, since a dataset represents a real-world entity as a single object. Second, only locations of objects are used to find the fusion sets. This assumption is feasible, since spatial objects always have information about their locations. Third, obviously, corresponding objects are within a distance D, but are not always the closest to each other, since locations are uncertain.

The two datasets are denoted as $A = \{a_1, \cdots, a_m\}$ and $B = \{b_1, \cdots, b_n\}$ respectively. Two objects $a \in A$ and $b \in B$ are corresponding objects, if they represent the same en-

tity. A fusion set that is generated from A and B is either a singleton (i.e., contains a single object) or has two objects, one from each dataset. A fusion set $\{a,b\}$ is correct if a and b are corresponding objects. A singleton fusion set $\{a\}$ is correct if a does not have a corresponding object in the other dataset.

In the absence of any global key, it is not always possible to find all the correct fusion sets. So, similarly to Beeri's work (Beeri et al, 2004), the quality of a fusion algorithm is measured in terms of recall and precision. **Recall** is the percentage of correct fusion sets that actually appear in result. **Precision** is the percentage of correct fusion sets out of all the fusion sets in the result. Formally, let the result of a fusion algorithm have N^r fusion sets, and let N_c^r sets be those that are correct. Let E denote the total number of real-world entities that are represented in at least one of the two datasets. Then the recall is N_c^r / E and the precision is N_c^r / N^r.

Factors affecting recall and precision are various. One factor is the **error interval**. The error interval is a bound on the distance between an object in the dataset and the entity it represents. The **density** of a dataset is the number of objects per unit of area. The **chosen object** is the number of objects in a circle with a radius that equals to the error interval. Apparently, the chosen object is the product of the density and the area of that circle. Intuitively, for a given entity, the chosen object is an estimate of the number of objects in the dataset that could possibly represent that entity. It is more difficult to achieve high recall and precision when the choice object is large.

Suppose that the datasets A and B have m and n objects, respectively. Let c be the number of corresponding objects. Then the number of distinct entities that are represented in the two datasets is m+n-(c/2). The overlap between A and B is c/(m+n). The overlap is a measure of the fraction of objects that have a corresponding object in the other set. One of the challenges one faces is to develop an algorithm that has high recall and precision for all degrees of overlap.

7.3 A Cell-Based Method Finding Fusion Sets

In this section, a cell-based method is proposed for solving the object-fusion problem. The method is based on the intuition that two corresponding objects must be within a distance D (it is practically the error interval D), but the two objects are not always the closest. The recall and precision of this method depend on specific characteristics of the given datasets, e.g., the chosen objects of the datasets, the degree of overlap, etc.

7.3.1 The Method

The four methods discussed in (Beeri et al, 2004; Minami, 2000) all depend on the distance between two objects in two datasets $A = \{a_1, \cdots, a_m\}$ and $B = \{b_1, \cdots, b_n\}$. We can call them distance-based methods. Computing distance between objects is an expensive and tedious task. So, the idea of the new method is to find fusion sets by using cell-by-cell processing instead of object-by-object processing, thereby avoiding the computation of distance between two objects. In addition, the method computes a confidence degree for every fusion set. The confidence degree indicates the likelihood that the fusion set is correct.

The final result is produced by choosing the fusion sets which confidence is above a given threshold. The threshold $\tau (0 \le \tau \le 1)$ is given by the user. Typically, increasing the threshold will increase the precision and lower the recall, while decreasing the threshold will increase the recall and decrease the precision. Controlling the recall and precision by means of a threshold is especially useful when the datasets have a large number of objects.

In the following, a special cell that is used in computing fusion sets is defined, and then the properties of this cell are discussed. For ease of presentation, suppose the data objects are 2-D (the case of higher dimensions will be the same). Each of the two data-sets A and B is quantized into cells or squares of length $l = \frac{D}{2\sqrt{2}}$ (This D is the error interval that is chosen by the user or obtained by a special process). Let $C_{x,y}$ denote the cell that is at the intersection of row x and column y. The layer 1 (L_1) neighbours of the cell $C_{x,y}$ are the immediately neighbouring cells of $C_{x,y}$, defined in the usual sense, that is,

$$L_1(C_{x,y}) = \{C_{u,v} \mid u = x \pm 1, v = y \pm 1, C_{u,v} \neq C_{x,y}\} \qquad (1)$$

A typical cell (except for cell on the boundary of the cell structure) has 8 L_1 neighbours.

Definition 7.1 Two objects a and b are called **the same cell objects**, if $a \in A$ and $b \in B$ lie within the same cell $C_{x,y}$.

Property 7.1 Any pair of the same cell objects is at most distance $\frac{D}{2}$ apart.

Definition 7.2 If $C_{u,v}$ is a L_1 neighbour of $C_{x,y}$, then any object $a \in C_{u,v}$ ($a \in A$) is **the L_1 neighbour object** with respect to any object $b \in C_{x,y}$ ($b \in B$).

The definition of $b \in B$ is the L_1 neighbour from $a \in A$ can be defined similarly.

Property 7.2 If $C_{u,v}$ is a L_1 neighbour of $C_{x,y}$, then any object $p \in C_{u,v}$ and any object $q \in C_{x,y}$ is at most distance D apart.

Property 7.1 is valid because the length of a cell's diagonal is $\sqrt{2}l = \sqrt{2}\frac{D}{2\sqrt{2}} = \frac{D}{2}$. Property 7.2 is valid too, because the distance between any pair of objects in two cells cannot exceed twice the length of a cell's diagonal. From these two properties, one can say that any pair of objects, which they are the same cell objects or one is the L_1 neighbour object of other's, may be a pair of fusion set.

The layer 2 (L_2) neighbours of $C_{x,y}$ are those additional cells within 3 cells of $C_{x,y}$, i.e.,

$$L_2(C_{x,y}) = \{C_{u,v} \mid u = x \pm 3, v = y \pm 3, C_{u,v} \notin L_1(C_{x,y}), C_{u,v} \neq C_{x,y}\} \quad (2)$$

A typical cell (except for any cell on or near a boundary) has 7^2-3^2=40 L_2 cells. Note that layer 1 is 1 cell thick and the layer 2 is 2 cells thick. L_2 chosen in this way satisfies the following property.

Definition 7.3 If $C_{u,v}$ is a L_2 neighbour of $C_{x,y}$, then any object $a \in C_{u,v}$ ($a \in A$) is the L_2 neighbour object with respect to any object $b \in C_{x,y}$ ($b \in B$).

The definition of $b \in B$ is the L_2 neighbour object from $a \in A$ can be defined similarly.

Property 7.3 If $C_{u,v} \neq C_{x,y}$ is neither a L_1 nor a L_2 neighbour of $C_{x,y}$, then any object $p \in C_{u,v}$ and any object $q \in C_{x,y}$ must be greater than the distance D apart.

Since the combined thickness of L_1 and L_2 is 3 cells, the distance between p and q must exceed $3l = \frac{3D}{2\sqrt{2}} > D$. The error interval D is a bound on the distance between an object in the dataset and the entity it represents, so the following definition can be given.

Definition 7.4 Any object $a \in C_{x,y}$ ($a \in A$) is an **isolated object** if and only if there is not any object $b \in B$ in the same cell, L_1 neighbour and L_2 neighbour of object a.

The definition of an isolated object of $b \in B$ can be defined similarly.

The intuition behind the cell-based method is that corresponding objects are within a distance D. So, in the cell-based method, a two-object fusion set is created for each pair of the same cell objects and the L_1 neighbour objects. A singleton fusion set is created for each isolated object.

Now, the degree of confidence of fusion sets will be defined. Consider a pair of the same cell objects or L_1 neighbour objects (i.e., $a \in A$, $b \in B$ and $a,b \in C_{x,y}$, or $a \in A$,

$b \in B$ and {$^{a \in C_{x,y}}$, $b \in L_1(C_{x,y})$ or $a \in L_1(C_{x,y})$, $^{b \in C_{x,y}}$}). Let $B_{count}(C_{x,y})$ be the number of B's objects in $C_{x,y}$, and $B_{count}(L_1(C_{x,y}))$ be the number of B's objects in L_1 neighbours of $C_{x,y}$. The definitions of $A_{count}(C_{x,y})$ and $A_{count}(L_1(C_{x,y}))$ are similar. The confidence degree of the fusion set $\{a,b\}$ is defined as follows.

$$confidence(\{a,b\}) = \sqrt{\frac{1}{B_{count}(C_{x,y}) + B_{count}(L_1(C_{x,y}))} \times \frac{1}{A_{count}(C_{x,y}) + A_{count}(L_1(C_{x,y}))}} \quad (3)$$

The confidence is defined as the square root in order for it to be not too small. If a is the only one of the A's object in $C_{x,y}$ and $L_1(C_{x,y})$ and b is the only one of the B's object in $C_{x,y}$ and $L_1(C_{x,y})$, then its confidence is the largest, and is equal to 1. The addition of the same cell objects or L_1 neighbour objects of A or B has reduced the confidence of the fusion set $\{a,b\}$.

Now consider an isolated object $a \in A$ ($^{a \in C_{x,y}}$), i.e., any $b \in B$ is not in the $C_{x,y}$ and L_1 of $C_{x,y}$. Let $B_{count}(L_2(C_{x,y}))$ be the number of B's objects in L_2 neighbours of $C_{x,y}$. The confidence degree of the fusion set $\{a\}$ is defined as follows.

$$confidence(\{a\}) = \sqrt{\frac{1}{1 + B_{count}(L_2(C_{x,y}))}} \quad (4)$$

The confidence will be the largest when the objects of $L_2(C_{x,y})$ is null, and is equal to 1. The confidence of a fusion set with a single object from $b \in B$ is defined similarly.

A threshold can be used to increase the precision of result by choosing only these fusion sets that have a confidence above the threshold. Consequently, some objects from given datasets may not be in the result. If the number of these objects is not so small, one have to perform again by using a smaller D, and again if necessary. For fewer objects, a less restrictive approach is to discard two-object fusion sets with a confidence below the threshold, but to add their objects as singletons.

In the method, the confidence of each cell is computed by using expression (3) or (4). When the confidence is above the threshold, the two objects in this cell or its L_1 neighbour cells will be paired randomly (Because two corresponding objects must be within a distance D, but the two objects are not always the closest).

The main advantage of the cell-based method over the traditional methods is efficiency and lower sensitivity to the degree of overlap between the two datasets. In particular, it may perform well even when neither datasets is covered by the other one, because a fusion set is paired randomly within a distance D.

7.3.2 The Algorithm

The algorithm of the cell-based method is designed as follows.

1). For $q \leftarrow 1,2,\cdots,m$, $Count_q^A \leftarrow 0, Count_q^B \leftarrow 0$

2). For each object $a \in A$ do

a. Map a to its appropriate cell C_q

b. Increment $Count_q^A$ by 1

3). For each object $b \in B$ do

a. Map b to its appropriate cell C_q

b. Increment $Count_q^B$ by 1

4). For $q \leftarrow 1,2,\cdots,m$ do

a. $Count_{q2}^A \leftarrow Count_q^A + \sum_{i \in L_1(C_q)} Count_i^A$

b. $Count_{q2}^B \leftarrow Count_q^B + \sum_{i \in L_1(C_q)} Count_i^B$

c. If $\sqrt{\frac{1}{Count_{q2}^A} \times \frac{1}{Count_{q2}^B}} > \tau$ do

For each object $a \in A$ in the C_q, find out an object $b \in B$ in this C_q or the L_1 of C_q at random and label $\{a,b\}$ as a fusion set, provided the b has not already been labeled.

d. If $Count_{q2}^B = 0$ or $Count_{q2}^A = 0$ do

i. $Count_{q3}^B \leftarrow \sum_{i \in L_2(C_q)} Count_i^B$, $Count_{q3}^A \leftarrow \sum_{i \in L_2(C_q)} Count_i^A$

ii. If $\sqrt{\frac{1}{Count_{q3}^B}} > \tau$ do

Label each object $a \in A$ and $a \in C_q$ as a singleton set $\{a\}$.

iii. If $\sqrt{\frac{1}{Count_{q3}^A}} > \tau$ do

Label each object $b \in B$ and $b \in C_q$ as a singleton set $\{b\}$.

5). If there are some objects that have not been labeled, change the size of the cell and repeat the 1-4 (note steps 2) and 3) only quantize these un-labeled objects to its appropriate cell at this time), until all objects are labeled.

Steps 2) and 3) of the algorithm quantize each object of dataset A and B to its appropriate cell. Step 4) labels all fusion sets and singleton sets of satisfying threshold τ. Step 5) repeats all the processes if D is set smaller, in order to deal with unlabelled objects.

7.3.3 Complexity Analysis

Let us analyze the complexity of this algorithm. Step 1) takes O(m) time, where $m<<N_A+N_B$ is the total number of cells (N_A and N_B are the number of objects in dataset A and B respectively). Steps 2) and 3) take O (N_A) and O (N_B) time respectively. Step 4) is the most complicated step. This step is the core of the algorithm. It divides N_A+N_B objects to m cells, and conquers them. In detail, the computing of 4(a), 4(b) and 4(d)i is fi-

nite, because a cell's neighbours, L_1 and L_2, are finite. For threshold τ, the number of A's objects and B's objects in C_q should be less than k_q^A and k_q^B respectively (k_q^A and k_q^B are constants). Therefore, O ($k_q^A + k_q^B$) time is required for each cell C_q (i.e., the time to perform an iteration of 4(c), 4(d)ii and 4(d)iii). Here, in the worst case, step 4) takes $O(\sum_{i=1}^{m}(k_i^A + k_i^B)) << O(N_A + N_B)$. For Step 5), if suppose the number of the loop iteration by Step 5) is *k* (this *k* will depend on the initial value of the D and the distribution of the data sets), and the complexity of following loop iteration will not exceed the first loop, therefore, the complexity of the algorithm is $O(k \times (m + N_A + N_B + \sum_{i=1}^{m} k_i^A + k_i^B)) \approx O(k \times (N_A + N_B))$.

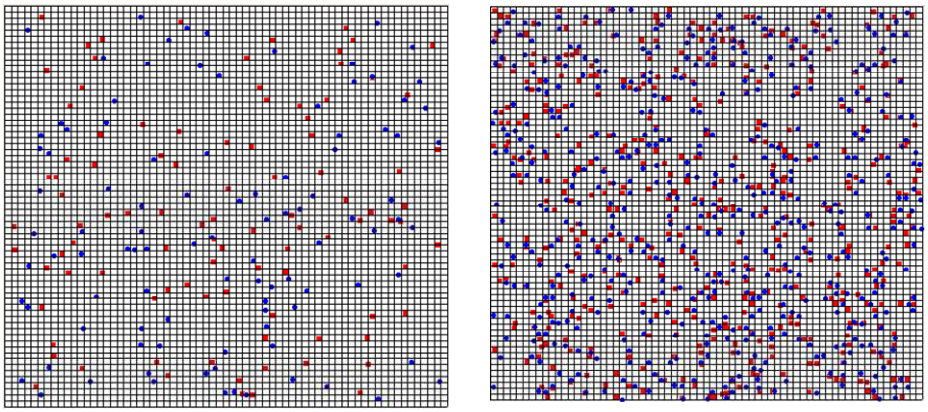
Note that the run-time will increase if D is reset in Step 5), although the complexity is unaltered. In Section 7.4, experimental results will show the efficiency of this algorithm.

7.4 Testing the Method

The method is tested using synthetic datasets, because there are not a sufficient number of real-world datasets to test our algorithm under varying degrees of density and overlap. Moreover, in real-world datasets, it is not always possible to determine accurately the correspondence between objects and real-world entities.

Following the method of the paper (Beeri et al, 2004), a synthetic dataset generator is implemented, which is a two-step process. First, the real-world entities are randomly generated. Second, the objects in each dataset are randomly generated, independent from the objects in the other dataset.

In the tests, 500 real-world entities is created in a square area of 2,000 * 2,000 meters, a minimal distance of 15 meters between entities, and an error interval of 30 meters for each dataset. And then three pairs of datasets were randomly generated. The three pairs had 100, 300, and 500 objects in each dataset, respectively (Figure 7.1 gives a visual view of the random pairs of datasets, with 100 and 500 objects). Thus, each pair had a different degree of overlap and density (500 objects in each dataset means a complete overlap).

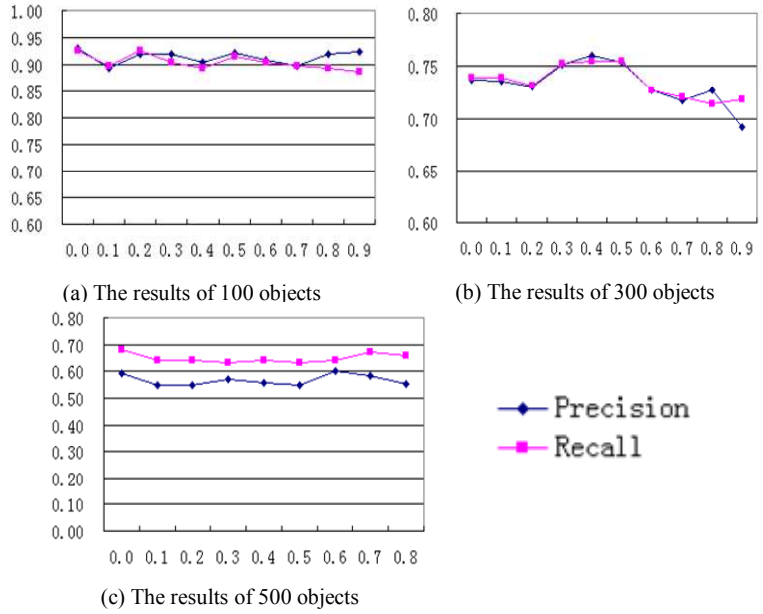


(a) 100 objects in each dataset (b) 500 objects in each dataset

Figure 7.1 A visual view of the random pairs of datasets, with 100 and 500 objects

The recall and the precision of the method are measured for every pair of datasets. In all tests, the initial value of D is 30 meters and the next iterative value of D is $D \times 0.9$. When the threshold τ was 0.0, for the datasets with 100 objects, the recall and the precision are both 0.93; for 300 objects, they are both 0.74, and for 500 objects, they were 0.68 and 0.59, respectively.

Figure 7.2 presents the recall and the precision as a function of the thresholds τ. From the tests, one can see that this algorithm is efficient and is insensitive to the degree of overlap between the two datasets. In particular, it may perform well even when neither dataset is covered by the other one (The values of the recall and the precision are all larger than 0.5), possibly because a fusion set is paired randomly within a distance D. One also can see that the recall and the precision have a low sensitivity to the threshold τ. This property will make the method useful in real applications.



(a) The results of 100 objects

(b) The results of 300 objects

(c) The results of 500 objects

Figure 7.2 Recall and precision as a function of the threshold values

Analyzing the effect of the initial value of the D on the performance of the algorithm, Figure 7.3 presents the recall and the precision as a function of the value of the D, and Figure 7.4 presents the running time as a function of the D. These tests tell us that the initial value D will have a significant effect on the results and efficiency of the algorithm (for example, the running time when D is set to 120 is more then two times of the value when D is set to 90. This means that the loop iteration *k* in initial D set to 120 is at least bigger than k′+2 (k′ is the number of the loop iterations when D is initially set to 90)). From Figure 7.4, the best results occurred when the initial value of D is the error interval (in these experiments, the threshold τ is set to 0.4).

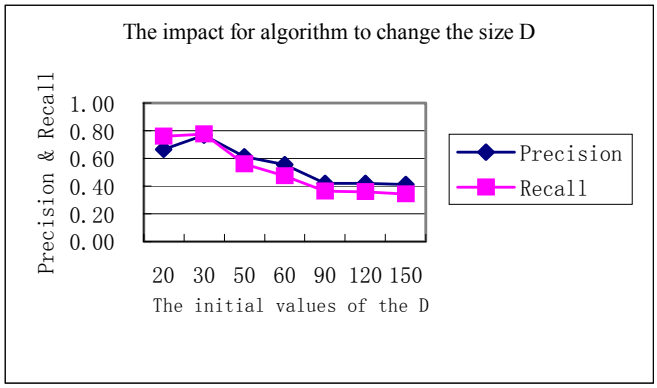


Figure 7.3 The impact for algorithm's precision to change the size of the D

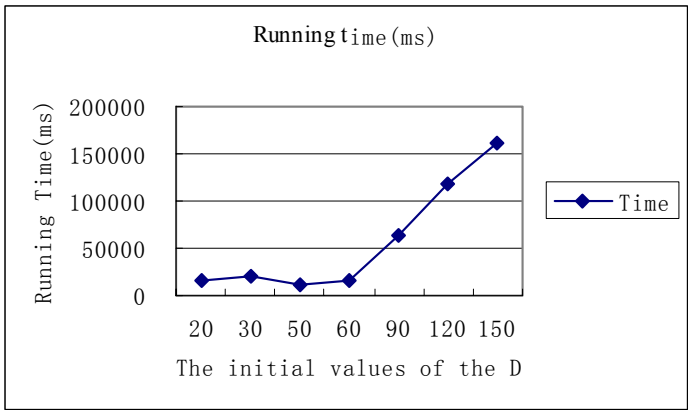


Figure 7.4 The impact for algorithm's time to change the size of the D

Figure 7.5 presents the impact for algorithm efficiency with changing the thresholds τ. From the tests, one can see that the running time (ms=millisecond) will increase when the threshold values τ is increasing, while the efficiency will be decreased when the overlap is increasing, since the number of iterations of the algorithm is increased. The thresholds (0.3-0.5) should be reasonable choices from Figure 7.5.

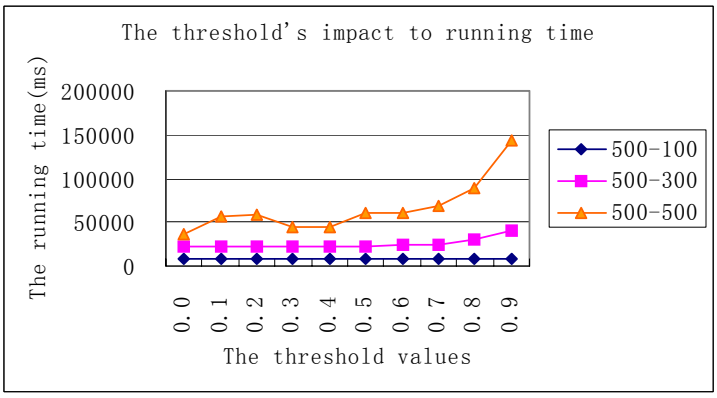


Figure 7.5 Running time as a function of the threshold values

7.5 Summary

A new approach to deal with the object fusion problem, based on a special cell with a length that is equal to the $\frac{D}{2\sqrt{2}}$ (D is the error interval), is developed. The key idea of this method is to find fusion sets by using cell-by-cell processing instead of object-by-object processing, thereby avoiding the computation of distance between two objects. Algorithm's measures of recall, precision and running time for the various degrees of overlap and thresholds are shown in extensive experiments.

The future work is to combine this approach with grid-clustering approach to spatial data mining (Wang et al, 1997; Agrawal et al, 1998; and Shi et al, 2003). It will be studied

how to utilize locations that are given as polygons (e.g., a mountain) or lines (e.g., a river), rather than just points.

In next Chapter, a fuzzy clustering method based on domain knowledge is discussed (Lu et al, 2007). In this method, the transitive closure is not computed by recursion, so the new algorithms save much time.

Chapter 8

A Fuzzy Clustering Method Based on Domain Knowledge

Clustering is an important task in data mining. Fuzzy clustering is on the significant status in clustering, which can deal with all types of datasets. The fuzzy clustering method in this chapter is based on domain knowledge, from which the tuples' semantic proximity matrix can be obtained, then two fuzzy clustering methods are introduced, which both started from semantic proximity matrix, so the results of fuzzy clustering can be instructed by domain knowledge. The two fuzzy clustering methods are Natural Method (NM) and Graph-Based Method (GBM), which are both controlled by a threshold that is confirmed by polynomial regression. Theoretical analysis testify the corrective of the new methods, the extensive experiments on synthetic datasets compare the performance of the new approaches with that of Modified MM approach in literature (Wang, 2000) and highlight the benefits of the new approaches, and the experimental results on real datasets discover some rules which are useful to domain experts.

8.1 Overview

Motivation: Clustering is an important task in data mining, which can discover useful information from plenty of datasets. But in practice, many types of datasets often need to be dealt with. In this chapter, the relationships from different plant species need to be discovered, the plant datasets are described by several data types, these kinds of problems are often met. So the fuzzy clustering methods are studied, which can deal with all types of datasets.

Related Works: Clustering has been studied extensively for 40 years and across many disciplines due to its broad applications. Most books on patterns classification and machine learning contain chapters on cluster analysis or unsupervised learning. Methods for combining variables of different types into a single dissimilarity matrix were introduced by Kaufman and Rousseeuw in (Kaufman and Rousseeuw, 1990). For partitioning methods, the k-means algorithm was first introduced by MacQueen in (MacQueen, 1967). The k-medoids algorithms of PAM and CLARA were proposed by Kaufman and Rousseeuw in (Kaufman and Rousseeuw, 1990). The CLARANS algorithm was proposed by Ng and Han in (Ng and Han, 1994). For density-based clustering methods, DBSCAN was proposed by Ester, Kriegel, Sabdae, and Xu in (Ester et al, 1996). The BIRCH algorithm was developed by Zhang et al in (Zhang et al, 1996). For fuzzy cluster-

ing methods are discussed in (Kaufman and Rousseeuw, 1990; Liu and Tian, 2001; Wang, 2000). These methods, however, are not efficient due to iteration.

Contributions: The contributions of the chapter are as follows. (1). In order to get the tuples' semantic proximity matrix, domain knowledge is used in fuzzy clustering. (2). introducing two clustering methods: Natural Method and Graph-Based Method, both of which are controlled by a threshold, and the threshold is confirmed by polynomial regression. (3). the experiments are on synthetic and real datasets respectively.

Organization: The rest of the chapter is arranged as below: In Section 8.2, the classical method for fuzzy clustering—Matrix Method is expatiated with an example. Introducing the two new methods, designing algorithms and confirming the threshold are all in Section 8.3. Section 8.4 discusses correctness and complexity of the new algorithms. Evaluation on experiments is in Sect. 8.5, the last is the summary.

8.2 Basic Concepts and Methods

Fuzzy clustering in this chapter has 3 steps:

(1). Reading all tuples in and replacing each attribute value by leaf concept in concept hierarchy tree;

(2). Computing every two tuples' semantic proximity and forming proximity matrix S according to domain knowledge—Concept Hierarchy Trees;

(3). Executing clustering using the proximity matrix S.

8.2.1 Basic Concepts

Concept hierarchy tree. Each attribute A_i ($1 \le i \le k$, k is the number of attributes) has a concept hierarchy tree H_i, which can be obtained from domain experts. The concept hierarchy trees of attributes "plant" and "elevation" in the plants dataset of "Three Parallel Rivers of Yunnan Protected Areas" are shown in Figure 8.1. The definitions of height, depth and level of node are the same with the Chapter 6 (see Definition 6.5).

It can be seen from Figure 8.1 that the higher level concepts are the generalization of lower level concepts, If the attribute's value is continuous, then leaf concepts are a interval with continuous values, one can use the method in literature (Kohavi and Sahanu, 1996; Tay and Shen, 2002) to discretize them, then deal with them according to discrete attribute values.

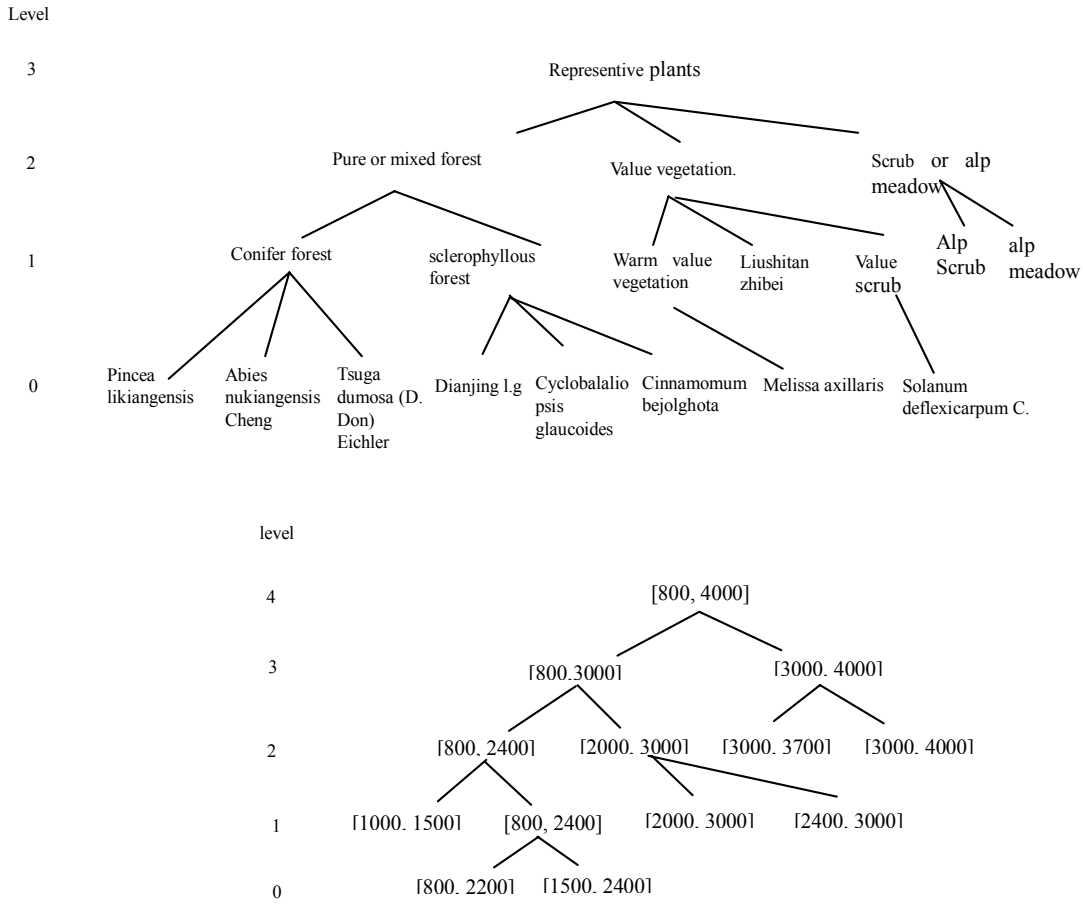


Figure 8.1. Concept hierarchy trees of attributes "plant" and "elevation"

Computing the tuples' proximity matrix S using concept hierarchy tree. Table 8.1 is parts of plants' dataset of "Three Parallel Rivers of Yunnan Protected Areas". At first, each attribute value in table 8.1 is replaced by corresponding leaf concept of concept hierarchy tree in Figure 8.1, and then the semantic proximity between attribute's values are computed using the equation (1) in definition 8.1.

Table 8.1 Plant and elevation datasets

Tuple ID	Plant	Elevation
t_1	Cinnamomum bejolghota	[1000,1500]
t_2	Tsuga dumosa (D. Don) Eichler	[2400,3000]
t_3	Abies nukiangensis Cheng	[3000,3700]
t_4	Melissa axillaris	[2000,3000]
t_5	Pincea likiangensis	[2000,3000]
t_6	Cyclobalaliopsis glaucoides	[3000,4000]
t_7	Dianjing l.g	[1500,2400]
t_8	Solanum deflexicarpum C.	[800,2200]
t_9	Tsuga dumosa (D. Don) Eichler	[2400,3000]
t_{10}	Liushitan zhibei	[2400,3000]

Definition 8.1. Attribute's Semantic proximity (**Attribute's proximity**, for short): For two leaf concepts in concept hierarchy tree f_1, f_2, the attribute's semantic proximity is defined as follows:

$$SP(f_1, f_2) = \begin{cases} 1, & f_1 = f_2 \\ 1 - l(P(f_1, f_2))/H & f_1 \neq f_2 \end{cases} \quad (1)$$

$P(f_1, f_2)$ returns the common father-node of f_1, f_2; and $l(f)$ returns the level of f; H is the Height of the tree. So the semantic proximity between attribute values can be computed using domain knowledge—concept hierarchy trees.

Definition 8.2. Tuples' semantic proximity (**Tuples' proximity**): Suppose there are two tuples $t_1 = (a_1, a_2, ..., a_k)$, $t_2 = (b_1, b_2, ..., b_k)$, the semantic proximity between tuples t_1, t_2 is defined as the equation (2).

$$SP(t_1, t_2) = \left[\sum_{i=1}^{k} W_i \times (SP(a_i, b_i))^2 \right]^{1/2} \quad (2)$$

Where W_i is the weight of attribute *i*.

Computing all tuples' proximity between tuples for a dataset (for example, Table 8.1) forms a proximity matrix S ($S = (s_{ij})_{n \times n}$, where $s_{ij} = 1$ *if* $i = j$; and $s_{ij} = sp(t_i, t_j)$ *if* $i \neq j$). S is reflexivity, symmetry but not transitive. So S is a similar matrix not equivalent matrix.

8.2.2 Fuzzy Clustering Using Matrix Method

Using matrix method (MM, for short) to realize fuzzy clustering has been introduced by Liu and Tian in (Liu and Tian, 2001). The process of the fuzzy clustering is: The similar matrix S is self-multiplied repeatedly, where $(s_{ij})^2 = MAX_k(MIN(s_{ik}, s_{kj}))$, until $S^{2k} = S^k$. S^{2k} is called as a fuzzy equivalence matrix (i.e., $s_{ij}^k = 1, s_{ij}^k = s_{ji}^k, S^{2k} = S^k$)

Example 8.1. According to concept hierarchy trees in Figure 8.1, the semantic proximity between tuples in table 8.1 form a similar matrix S shown in Figure 8.2(a) (Supposed weight are both 1/2):

$$S = \begin{bmatrix} 1 & 0.29 & 0.24 & 0.18 & 0.29 & 0.47 & 0.59 & 0.35 & 0.29 & 0.35 \\ 0.29 & 1 & 0.47 & 0.35 & 0.59 & 0.24 & 0.29 & 0.18 & 1 & 0.71 \\ 0.24 & 0.47 & 1 & 0 & 0.47 & 0.29 & 0.23 & 0 & 0.47 & 0 \\ 0.18 & 0.35 & 0 & 1 & 0.71 & 0 & 0.18 & 0.29 & 0.35 & 0.42 \\ 0.29 & 0.59 & 0.47 & 0.71 & 1 & 0.23 & 0.29 & 0.18 & 0.59 & 0.35 \\ 0.47 & 0.24 & 0.29 & 0 & 0.23 & 1 & 0.47 & 0 & 0.23 & 0 \\ 0.59 & 0.29 & 0.23 & 0.18 & 0.29 & 0.47 & 1 & 0.53 & 0.29 & 0.18 \\ 0.35 & 0.18 & 0 & 0.29 & 0.18 & 0 & 0.53 & 1 & 0.18 & 0.29 \\ 0.29 & 1 & 0.47 & 0.35 & 0.59 & 0.23 & 0.29 & 0.18 & 1 & 0.71 \\ 0.35 & 0.71 & 0 & 0.42 & 0.35 & 0 & 0.18 & 0.29 & 0.71 & 1 \end{bmatrix}$$

(a). the similar matrix S

$$S^4_{0.44} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

(b). the level value matrix $S^4_{0.44}$

Figure 8.2. An example of fuzzy clustering process

The similar matrix *S* multiplies itself repeatedly, an equivalent matrix S^4 ($S^4 = S^8$, so S^4 is the equivalent matrix) can be obtained. The clustering result can be obtained by setting a threshold. If the threshold λ=0.44, the level value matrix $S^4_{0.44}$ is obtained as Figure 8.2(b) (the value becomes 1 if it is greater than λ, otherwise zero). The clustering result is $T_{0.44} = \{(t_1, t_6, t_7 t_8), (t_2, t_3, t_4, t_5, t_9, t_{10})\}$, and if setting the threshold is $\lambda = 0.48$, the result is $T_{0.48} = \{(t_1, t_7 t_8), (t_2, t_4, t_5, t_9, t_{10}), (t_3), (t_6)\}$.

A modified matrix method (MMM, for short) is presented in Chapter 2, and the main modified idea is that it does not deal with the value “0” in similar matrix (i.e., to avoid meaningless looping) and saves running time, especially when the number of 0 is very large. The new methods of this chapter also start from the similar matrix *S*.

8.3 New Algorithms for Fuzzy Clustering

8.3.1 Natural Method (NM)

It is from a simple idea, which directly uses the similar matrix S to process clustering. At first, saves the nonzero elements (proximities) and their rows, columns in up-triangular of the similar matrix in three arrays a[], v[] and w[], then scans the three arrays to get clustering result. The method is described as below:

If two tuples (that is v[i] and w[i]) have not been searched and the proximity (a[i]) is bigger than the threshold λ, then putting them in a new category;

If only one tuple has been searched, and proximity is bigger than λ, put the tuple which has not been searched into the category possesses the other tuple.

If two tuples have been both searched and the proximity is bigger than λ, then combining the two categories which the two tuples belonged to into one.

So, the clustering result can be obtained by scanning three arrays a[], v[] and w[] once. The algorithm is given in following.

Algorithm 8.1 `Natural Method for Fuzzy Clustering (NM)`

Input: the nonzero elements (proximities) and their rows, columns in up-triangular of similar matrix a[], v[], w[]

Output: classifications

Method:

```
(1) For i:=1 to n do //n is the number of tuples
      Cla[i]:=0;    //the classification values of tuple i are initialized to 0
    t:=0;
(2) For i:=1 to m do  // m is the number of the nonzero elements in similar matrix S
(3)   If a[i]>= λ then
      {
(4)     If Cla[v[i]]=0 and Cla[w[i]]=0 then
          {t:=t+1; Cla[v[i]]:=t; Cla[w[i]]:=t}
(5)    Else  if (Cla[v[i]]=0 and Cla[w[i]]<>0) then
                 Cla[v[i]]:= Cla[w[i]]
               Else if (Cla[v[i]]<>0 and Cla[w[i]]=0) then
                 Cla[w[i]]:= Cla[v[i]]
(6)                Else  //the two tuples have been scanned
                     Changing the classification values of the tuples that they has the
                      classification values Cla[v[i]] and Cla[w[i]] into the same one;
      };
```

8.3.2 Graph-Based Method (GBM)

A graph (G (V, E)) can be stored by a matrix or adjacent table. It is expressed with an adjacent table here. Taking the tuples as vertexes of graph, and the proximity is as the weight of arcs. The table head *Header* is a dynamic array, which stores the tuple id of each tuple in sequence. Each node of links has three fields: node.col, node.value and node.next, and they store column, proximity and pointer points to the next node respectively. When constructing adjacent table, the arcs which's weights smaller than λ can be directly deleted (the proof will be in section 8.4). Therefore, a collection of non-connective sub-graphs, these sub-graphs are equivalent partition to original graph (see the proof in section 8.4), is obtained.

The algorithm is as follows:

Algorithm 8.2. `The Graph-Based Method for Fuzzy Clustering (GBM)`

Method: 1) Create( ); // Construct adjacent table only to the arc whose weight is bigger than λ

2) Depth_Scan(1,1); // Start from the first element of Header, execute depth first search to the adjacent table

3) Execute depth first search to the node that has not been searched, and stopping search until all nodes have been searched.

The step 2）in algorithm 8.2 only scanned (searched) one sub-graph, and searched all sub-graphs in step 3). The procedure Create() is designed as below.

```
Procedure Create()
 (1) for i:= 1 to m do    //m the number of the nonzero elements in similar matrix S
     begin
 (2)    new node node1;
 (3)    new node node2;
        node1.col := w[i];   node1.value := a[i];   node1.next := null;
        node2.col := v[i];   node2.value := a[i];   node2.next := null;
 (4)    if a>λ then
        begin
 (5)      if (Header[v[i]].next=null)            //to insert the node1
            Header[v[i]].next:=node1
 (6)      else
            insert the node1 into the header[v[i]] link;
          if (Header[w[i]].next=null)            // to insert the node2
            Header[w[i]].next:=node2
          else
            insert the node2 into the header[w[i]] link;
          end;
      end.
```

8.3.3 Confirming the Threshold λ

Carefully analyzing Matrix Method you will find, it implements equivalent partition according to λ after having obtained equivalent matrix, so you can change λ to get a reasonable result. While the threshold λ is used in the beginning of the new algorithms, the whole clustering process has to be performed again if λ changed. Therefore, confirming the threshold λ is a very important task in the new methods. Here confirms λ by polynomial regression, which is enlightened by the method in (Zhang et al, 2004). The main steps are as below.

(1). Confirming a λ by Polynomial Regression. Arraying the nonzero elements (proximities) in up-triangular of similar matrix by descent, then executing thrice-polynomial regression to points who's Y-coordinates are these proximities, the inflexion of curve is the threshold λ.

(2). Computing the nearest k entries values from the former λ and choosing a λ**.** Computing the nearest k entries values from the λ obtained in step (1) and saving them, which are a possible scopes of final λ, then choosing a given λ from the scopes to perform clustering according to the need of user.

The matrix method has not any indication to confirm λ, so user can only try one by one, while the new algorithms firstly confirm a possible scope of threshold λ, user can choose λ from the scope. If it is not reasonable, one can choose over again from the scopes, and two clustering algorithms in this chapter are not time-consuming ($O(n*m)$), which will be better than trying one by one in practice (It will be shown by experiments in Section 8.5.1).

8.4 Algorithm Analysis

8.4.1 Correctness Analysis

Illustration. For the similar matrix S in example 8.1, the fuzzy clustering is executed using GBM with the threshold $\lambda = 0.44$ and $\lambda = 0.48$. Figure 8.3(a), Figure 8.3(b) shows the sub-graphs after deleting the arcs whose weights are smaller than λ. After depth first search, the partition results can be obtained, which are the same as the results of the MM. The NM can obtain the same results too.

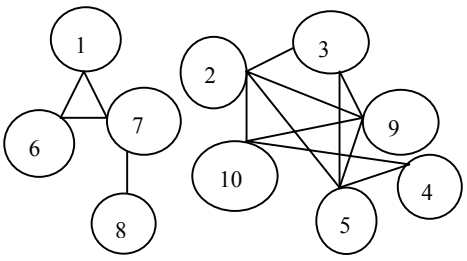


Figure 8.3(a) Sub-graph when $\lambda = 0.44$

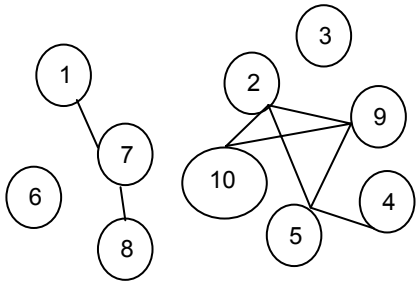


Figure 8.3(b) Sub-graph when $\lambda = 0.48$

Theoretical analysis. Analyzing the Matrix Method or Modified MM in section 8.2, the following rules will be found:

①The matrix multiplies itself in fact computes the transitive closure.

②Given a threshold λ, the values smaller than λ can be replaced by zero, which cannot affect the equivalent matrix .

Proof. Supposed $S[i,j]$ and $\overline{S}[i,j]$ are values in similar matrix S and equivalent matrix $\overline{S}$ respectively, $\overline{S}[i,j]$ can be obtained by performing the following compution from $S[i,j]$:

$$\overline{S}[i,j] = \max(\min_k(S_{ik}, S_{kj})) = \max(..., \min(S_{il}, S_{lj}), ...)$$

$= \min(S_{il}, S_{lj})$ (suppose S_{il} and S_{lj} are the maximal values currently)

So $\overline{S}[i,j] = \min(S_{il}, S_{lj})$ may be two kinds of values: bigger than λ, smaller than or equal to λ.

① $\overline{S}[i,j] = \min(S_{il}, S_{lj}) > \lambda$,whether $S[i,j]$ is zero or not, $\overline{S}[i,j]$ will replace $S[i,j]$

② $\overline{S}[i,j] = \min(S_{il}, S_{lj}) \le \lambda$,whether $S[i,j]$ is zero or not, the value which is smaller than or equal to λ will be replaced by zero in the level value matrix.

③As one has known, computing the transitive closure to a graph is adding to some arcs. The arcs are defined as follows: If node 1 and node 2 are connected by an arc, node 2 and node 3 are connected by an arc, then add an arc between node 1 and node 3.

In this chapter, although GBM does not compute the transitive closure by recursion, but the set of non-connective sub-graphs is gotten after getting rid of some arcs. Even if computing transitive closure, the non-connective sub cannot become connective, the transitive closure is in fact only executed in each sub-graph, which will not affect clustering result.

8.4.2 Time Complexity

Comparing with MMM, all of the new algorithms need confirm λ at first. Although the computation of confirming will cost some time, the running time is fixed with use thrice-polynomial regression. Only the time of sorting will increase with the increasing of datasets, so the time complexity of confirming λ is $O(m * \log m)$. Table 8.2 shows the time complexity of the three algorithms. From Table 8.2 one can see NM and GBM are obvious better than MMM.

Table 8.2. The comparison of time complexity

Algorithm	Time Complexity
MMM	$O(n^3)$
NM	$O(n*m + m*\log m)\ (n < m)$
GBM	$O(n*m + m*\log m)$

8.5 Experiments

The performance of the new algorithms was evaluated with synthetic datasets and real-world datasets. The experiments were performed on a Celeron computer with a 2.60 GHz CPU and 512 Mbytes memory running the Windows XP operating system.

8.5.1 Evaluation Using Synthetic Datasets

The experiments using synthetic datasets want to answer the following questions. (1) How does the size of the dataset affect the three algorithms? (2) How is the three algorithms' performance with changing λ?

The experiments are executed on the matrix of 50×50, 100×100,…,and 500×500, comparing the performance of MMM, NM and GBM as function of matrix size respectively (see the Figure 8.4). When the matrix size is larger than 300, MMM cannot process due to lack of memory. From Figure 8.4 one can see that the new algorithms are better than MMM.

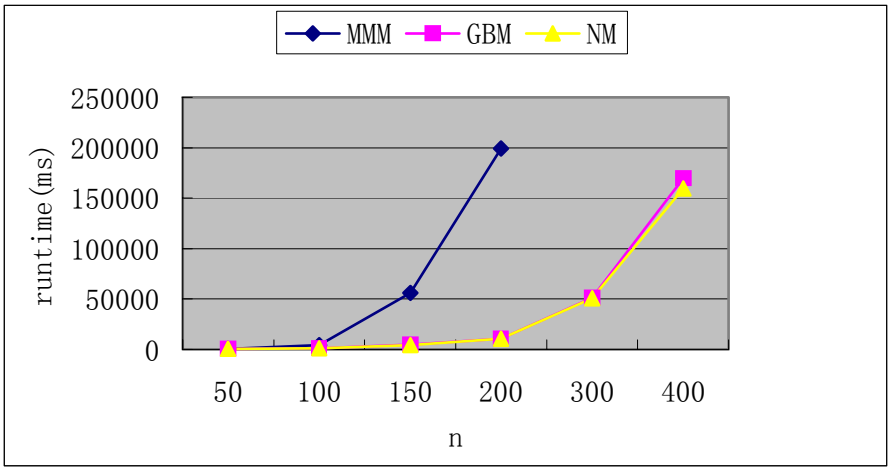


Figure 8.4. The performance of MMM, NM and GBM as function of matrix size n

With the fixed size of datasets, comparing the performance of NM with MMM by changing λ untill the reasonable clustering result gained. The experiments are executed on 100×100 matrix, table 8.3 shows the results. Yet, the experiments are only one example. From table 8.3 one can see that, because the cardinal number of MMM (the cost of first time) is very large, to get a reasonable result it must adjust thresholds λ many times, which means the total running time of MMM is larger than that of NM, and the performance of MMM declines very sharply with the increasing size of datasets.

Table 8.3. Runtime and total time of NM and MMM by changing λ

NM	λ	0.37	0.4	0.54	0.6			Total
	Runtime	894	48	46	40			1028
MMM	λ	0.1	0.2	0.3	0.4	0.5	0.6	Total
	Runtime	4281	28	32	24	20	25	4410

8.5.2 Evaluation Using a Real Dataset

The performance of the algorithms is evaluated using a real dataset. Can they discover useful information?

The real dataset used in experiments involves 16 plant species in “The Three Parallel Rives in Yunnan Areas”. There are 254 instances extracting from satellitic telemetric data. Some rules discovered by the algorithms are really interested by geographers and botanists. The followings are some examples:

Figure 8.5(a) is one of figures which have dealt with using some methods, which shows the 16 plant species with different colours respectively, and Figure 8.5(b) shows the clustering results using the new algorithms. Some rules discovered by the algorithms are useful. For example: Cordyceps sinensis (Berk.) Sacc is a kind of plant coloured with orange in Figure 8.5(a), but in Figure 8.5(b), they are divided in 3 categories with different colours respectively, which is due to the different growth environments, which explains the three kinds of environments are suitable for Cordyceps sinensis (Berk.) Sacc in “The Three Parallel Rives in Yunnan Protected Areas”.

On the other hand, some of Pseudotsuga forrestii Craib and Taxus wallichiana are in the 2-th category, which grow in the Cool temperate conifer forest whose elevation is from 3000 to 3400 meter of Lijiang (showed in Figure 8.5(b)), which explains these plant species can coexist in the same environment, and they may have some common characteristics. From plant hierarchy one can see that these plant species are all belonged to Gymnospermae, so domain knowledge validates the efficiency of the new algorithms.

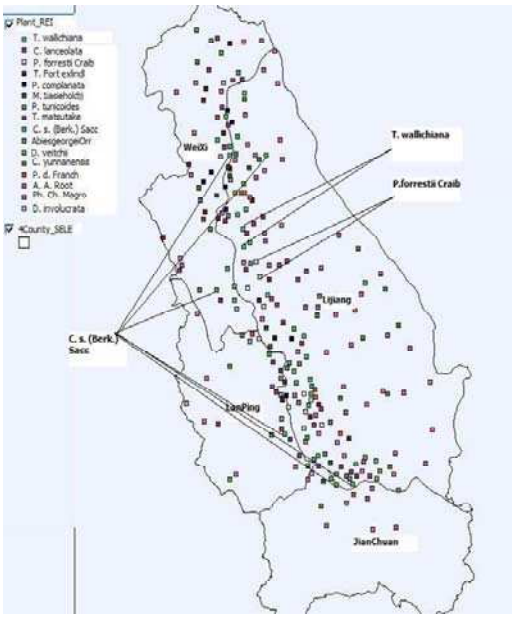


Figure 8.5(a).Before clustering

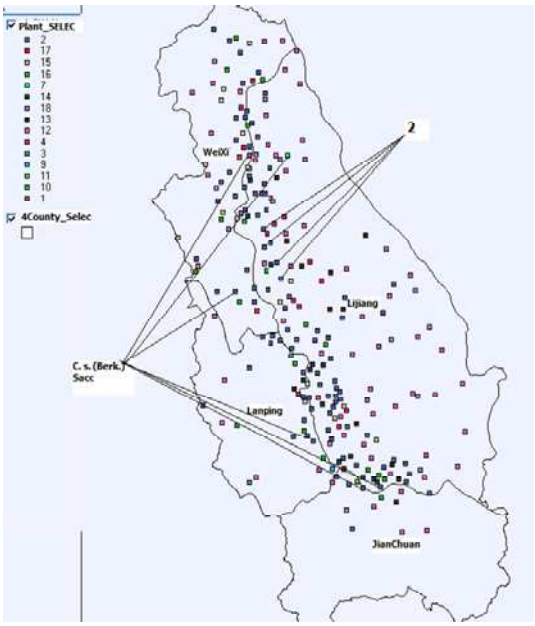


Figure 8.5(b). After clustering

8.6 Summary

The two fuzzy clustering methods—Natural Method and Graph-Based Method in this chapter are based on domain knowledge. After confirming threshold λ at the beginning of executing the new algorithms, the transitive closure does not need be computed by recursion and correct clustering results can be obtained, so the new algorithms save much time. The theoretical analysis and experimental results on synthetic and real datasets show that the new approaches are corrective and efficient, and some rules discovered by the new algorithms are useful to domain experts.

In experiments, it is found that the performance of algorithms will decline with the increasing of datasets. Further more, because GBM stored the whole adjacent table, when the experiment is executed on 500×500 matrix, it cannot be performed due to lack of memory. Therefore, how to solve the memory problems and realize data exchange between memory and storage are the future works.

A number of new techniques in the area of spatial data mining have been developed in previous chapters. These techniques can be incorporated into one software package, so that efficiencies and effect can be compared. In next Chapter, as an example, a prototype system of visual spatial co-location patterns mining is developed. In this system, the new techniques in Chapter 3 and 4 can be compared in any way you like.

Chapter 9

A Visual Spatial Co-location Patterns' Mining Prototype System (SCPMiner)

This chapter introduces a visual spatial co-location patterns' mining prototype system (SCPMiner). Visual spatial data mining is an effective way to discover knowledge from huge amounts of spatial data. The systematic study and development of visual spatial data mining techniques will facilitate the promotion and use of spatial data mining as a tool for spatial data analysis.

9.1 Overview

The purpose of studying spatial data mining is to support and improve spatial data-referenced decision-making in the real world. To reach this, the development of spatial data mining products and efforts toward the visualization studying is a very important researching direction. Although spatial data mining is a relatively new field with many issues that still need to be investigated in depth, some mining methods, for example, spatial co-location patterns mining, have been researched fully in this thesis. As a mining technique possessing broadly applied values, Research and development of a visual spatial co-location patterns' mining prototype system (SCPMiner) will facilitate the broad application of spatial co-location patterns' mining technique.

Mentioning the development of spatial data mining prototype system, there are the **GeoMiner** (Han and Kamber, 2006; http://db.cs.sfu.ca//GeoMiner) and the **MultiMediaMiner** (Zaiane et al, 1998) developed by Simon Fraser University in Canada, and the **RSImageMiner** developed by Wuhan University in China (Li et al, 2006). These systems have little in common with respect to data mining functionality or methodology, may including association mining, classification, prediction, clustering, outline detection, et al, but they work with different kinds of data sets and could not be used in spatial co-location patterns' mining.

9.2 Analysis and Design of SCPMiner

Spatial data mining functions form the core of a spatial data mining system. In the SCPMiner, only one data mining function, spatial co-location patterns' mining, is provided. But the extensible property is considered in designing of SCPMiner, it can become a system supported multiple spatial data mining functions. Because the methods

of co-location mining have advantages respectively for different kinds of data, the SCPMiner that supports multiple methods of the spatial co-location mining provide the user with greater flexibility and analysis power. Thus SCPMiner should also provide novice users with convenient access to the most suitable method, or to default settings. Figure 9.1 shows the basic architecture of SCPMiner.

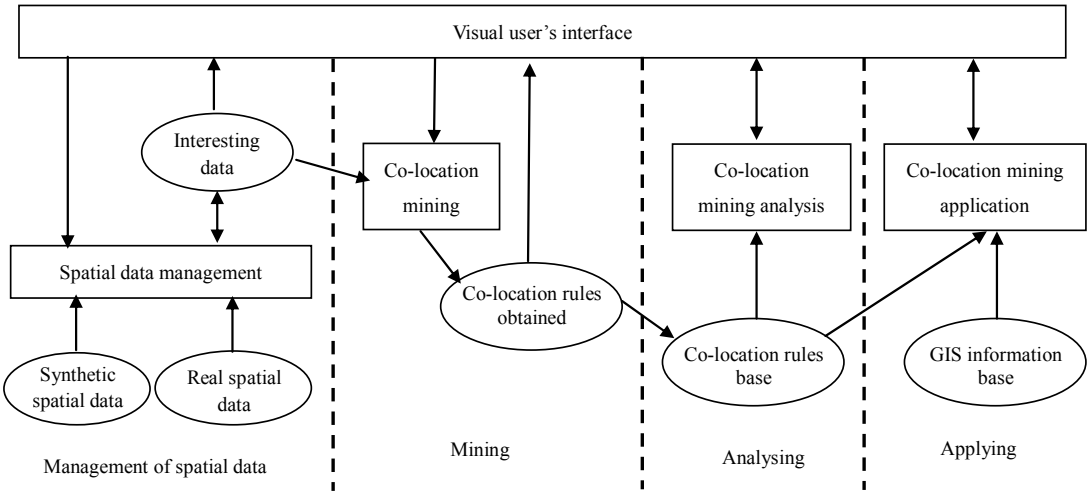


Figure 9.1 Basic architecture of SCPMiner

SCPMiner is divided into four parts in Figure 9.1. First (left side) is the management of co-location spatial data, second is the discovery of spatial co-location rules, third is co-location mining methods' analysis, and the fourth (right side) is the applications of co-location mining.

Before mining co-location knowledge, users could select interesting data from spatial dataset (synthetic or real) by using the **co-location data management procedure**. This is an interactive and visualization process. **Co-location mining procedure** accepts a command to mining co-location from user. According to the user's demand of knowledge mining, the co-location mining procedure discover knowledge from the interesting data. The co-location rules obtained is provided to user or added into the co-location rules base for users to query, analyze and apply them. **Co-location mining analysis procedure** includes efficiency analysis of mining methods and the characteristic analysis of mining data. Some application functions of co-location rules discovered are provided by **co-location mining application procedure**. In general, co-location mining process needs to be performed repeatedly to get satisfied results.

9.3 Implementation of SCPMiner

According to the designed architecture of SCPMiner in Figure 9.1, the SCPMiner was developed, under the operating system Window XP, by using JDK 1.4.2_5, Eclipse 3.2.2, adobe SVG Viewer tool, Macromedia Dreamweaver MX2004, and SQL Server2000 database management system. The main interface of SCPMiner is shown in Figure 9.2.

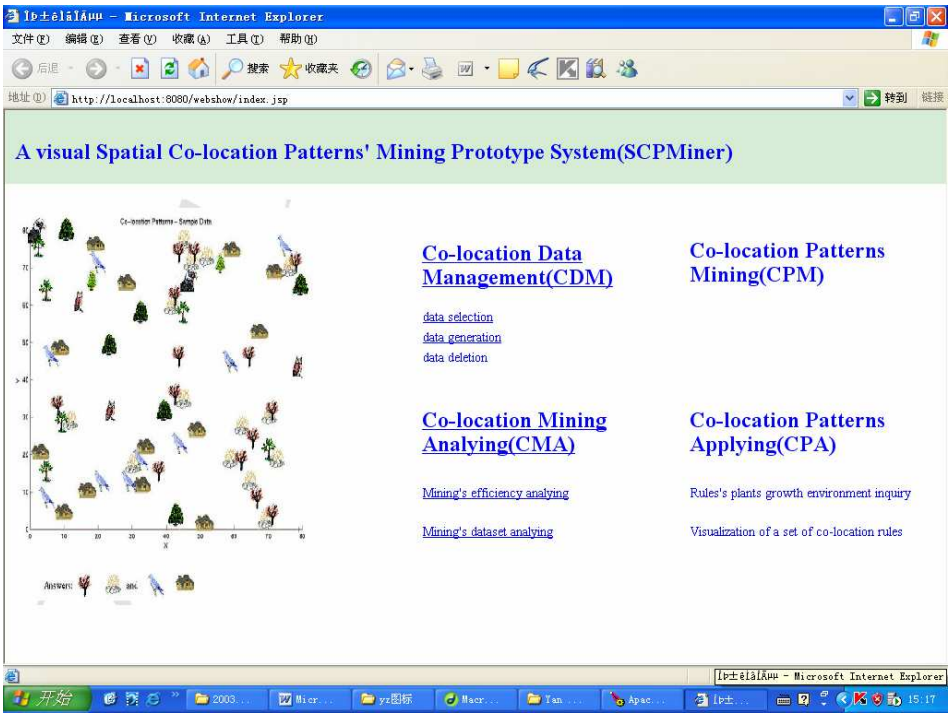


Figure 9.2 Main interface of SCPMiner

It can be seen that there are four main menus in SCPMiner. They are co-location data management, co-location rules mining, co-location mining analysis, and co-location mining applying.

9.3.1 Co-location Data Management (CDM)

The unitive management of spatial data is a feature of the SCPMiner. There are two kinds of spatial data in SCPMiner. One is synthetic spatial, another is real-world spatial data. One can browse the spatial data stored in the system, generate new synthetic data and delete data from the system. For real-world spatial data, there is a plants' distribution data of the "Three Parallel Rivers in Yunnan Protected Areas" in the system. One can load new real-world spatial data by CDM. Spatial data is stored in link structure. The real plants' distribution data of the "Three Parallel Rivers in Yunnan Protected Areas" is managed by SQL server 2000 database.

The interface of CDM is shown in Figure 9.3.

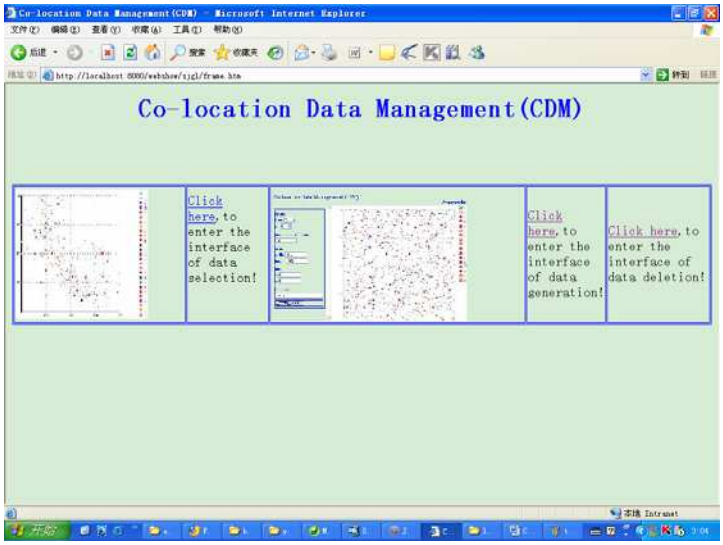


Figure 9.3 The interface of CDM

The main functions in CDM include:

(1) **Data selection**: selecting spatial data from exited spatial data set (synthetic or real) for co-location mining. The processing of selecting data is visual which means one can see the distribution of the data you want to select. Figure 9.4 is a result in the process of data selection. The selected spatial data will be stored in a data file for co-location mining.

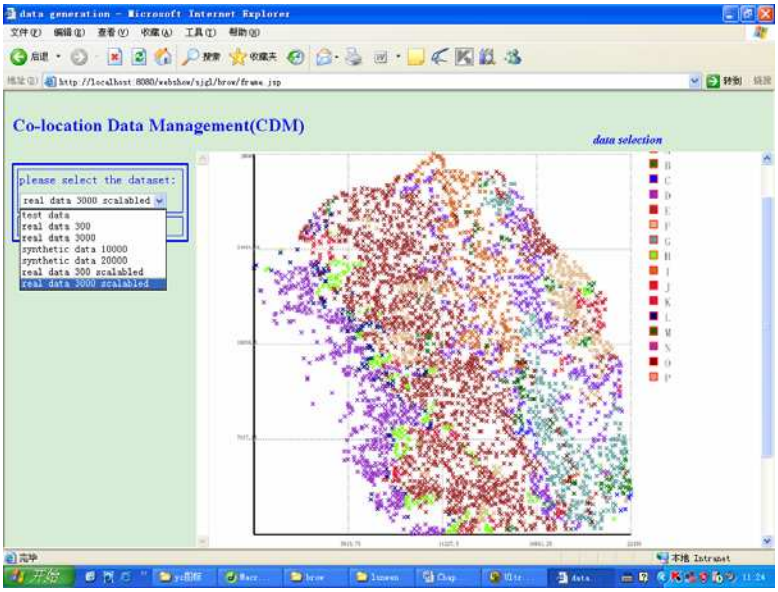


Figure 9.4 A result of selecting a plant distribution dataset

(2) **Data generation**: including synthetic spatial data generation and real spatial data input. The left side in Figure 9.5 is for setting parameters area. After setting suitable parameters, one can run the procedure of synthetic data generation. If you satisfy the result of data generation (the distribution of generated data is shown in right side area as showing the right side area of Figure 9.5), one can give a name to it and store it. For inputting real spatial data, the system can read file data and image data.

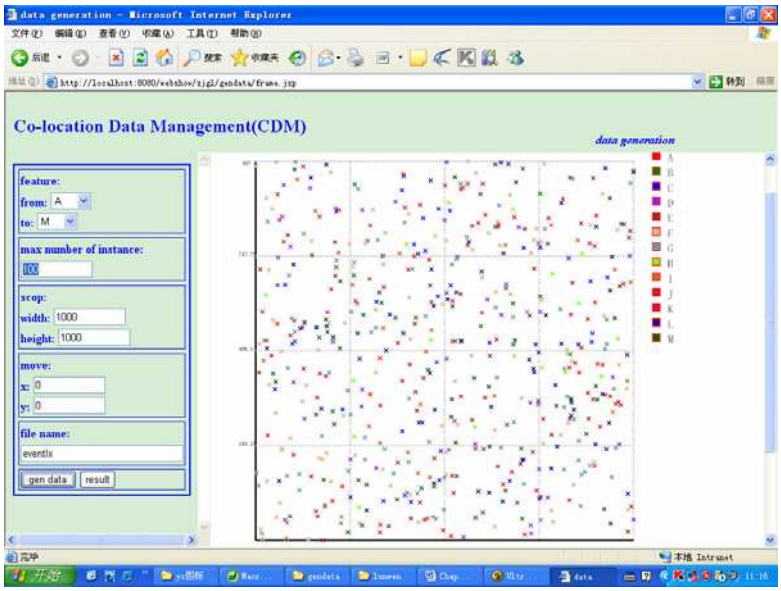
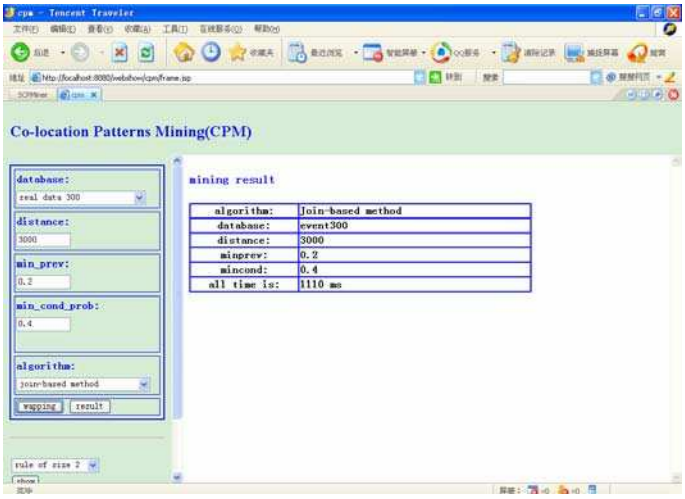


Figure 9.5 Processing of data generation

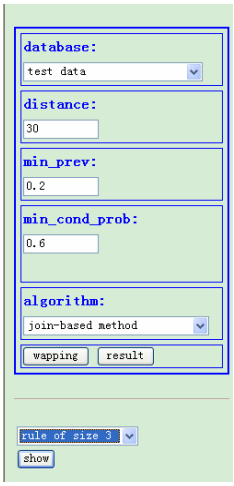
(3) **Data deletion**: deleting a selected synthetic spatial data or real spatial data from the system.

9.3.2 Co-location Patterns Mining (CPM)

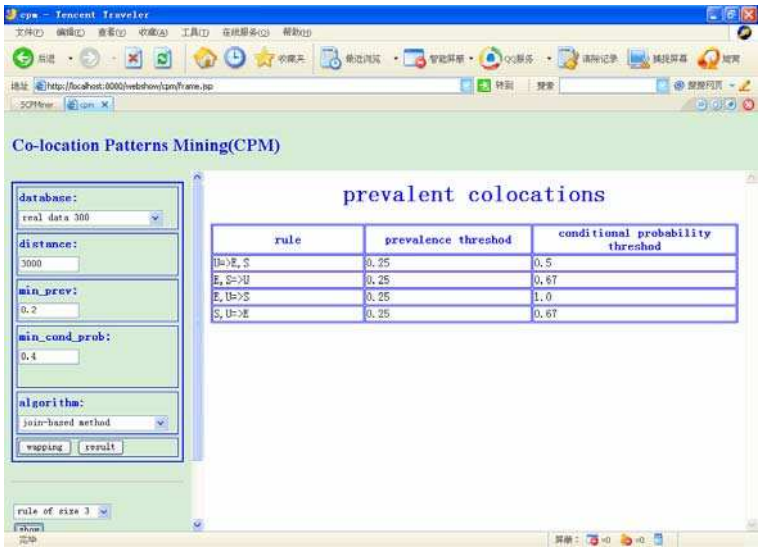
Figure 9.6(a) is the interface of CPM. The parameters of co-location patterns' mining include spatial neighbour distance *D*, minimum prevalence threshold *min_prev,* and conditional probability threshold *min_cond_prob*. The system provides the inter-interface (as shown in Figure 9.6(b)) to user to select values of these parameters. The scope of each parameter is computed according to the interesting data for co-location mining. CPM provide four algorithms for co-location mining: **join-based method** proposed by Huang et al (Huang et al, 2004), **join-less method** proposed by Yoo et al (Yoo et al, 2005), **CPI-tree method** proposed in this thesis, and **order-clique-based method** also proposed in here. One can choose any one of them to mining co-locations. But for different interesting data and different mining parameters setting, the system can guide you to choose a suitable method to mining co-locations. Figure 9.6(c) is a mining result of the size-3 co-location rules.



(a) The interface of CPM



(b) The interface of setting parameters



(c) A result of CPM

Figure 9.6 The procession of co-location patterns mining

9.3.3 Co-location Mining Analyzing (CMA)

In four co-location mining methods provided by the SCPMiner, different method represents its advantage under different spatial dataset. So, in CMA, two functions are provided for analyzing mining methods. One is **mining's efficiency analyzing**, and another is **mining's datasets analyzing**. Figure 9.7 is the interface of CMA.

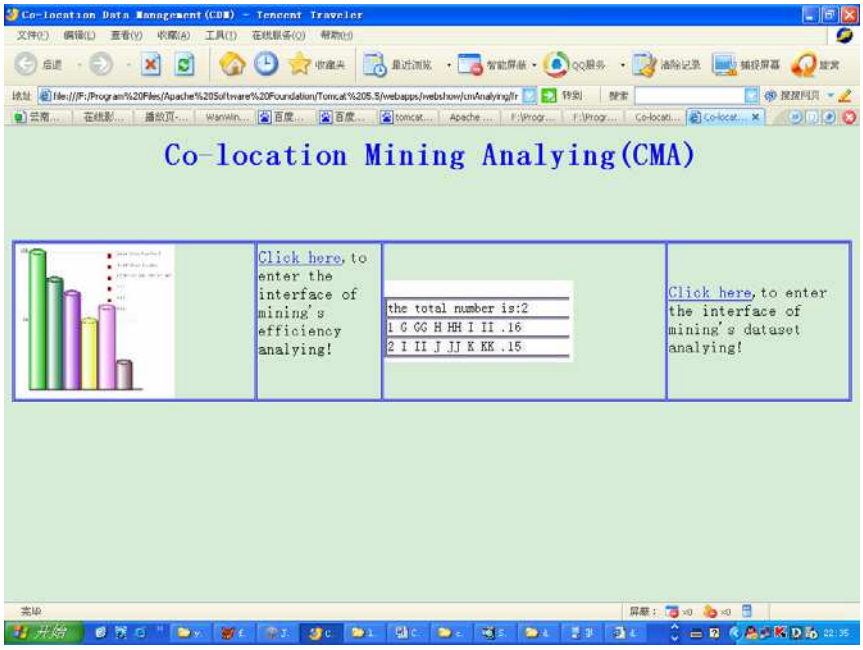


Figure 9.7 Interface of CMA

(1) **Mining's efficiency analyzing**: In this function, first, you select a dataset (sparse dataset or dense dataset). Second, you choose methods you want to compare. Third, spatial neighbour distance *D*, minimum prevalence threshold *min_prev,* or the number of instances *N* can be chosen one to analyzing efficiency of the mining method. Then, running results with *D, min_prev* or *N* over a dataset you selected will be obtained. Figure 9.8 is one result.

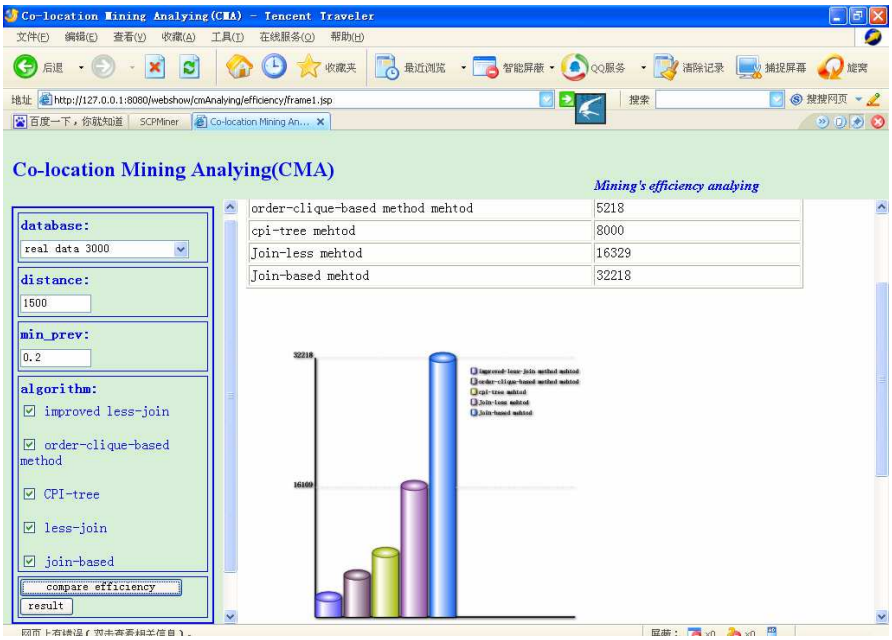


Figure 9.8 An example of mining's efficiency analysis

(2) **Mining's dataset analyzing**: If the analysis results above are not what you expected, you can analyze the dataset using this function. You can look at how distribution of this dataset, how many size-2 prevalence co-location patterns in this dataset. How long the maximal prevalence co-location pattern is in this dataset, and et al. The results of the size-2 prevalence co-location patterns in the dataset of Figure 9.8 are shown in Figure 9.9.

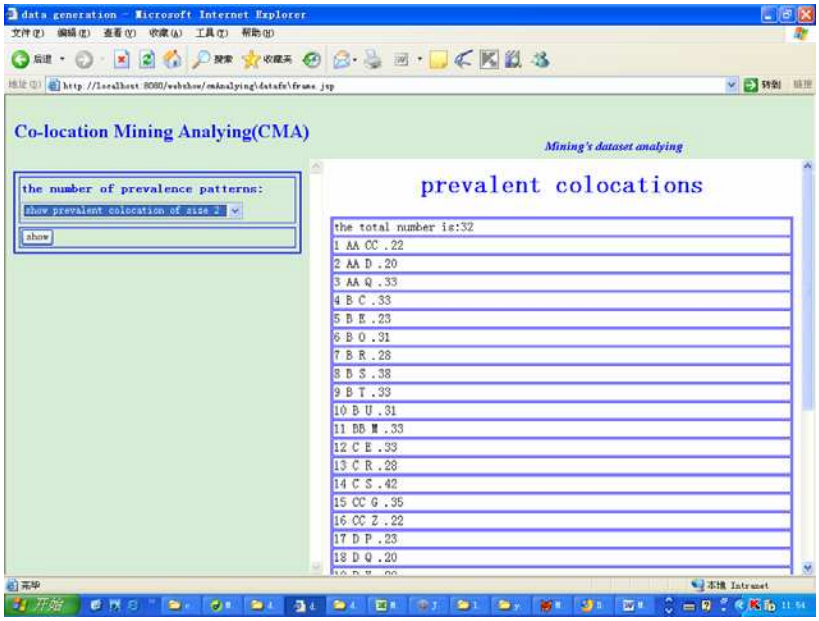


Figure 9.9 The results of the size-2 prevalence co-location patterns in the dataset of figure 9.8

9.3.4 Co-location Patterns Applying (CPA)

For there is a real plants distribution dataset of the "Three Parallel Rivers of Yunnan Protected Areas", the CPA provide two functions, **rules' plants growth environment query** and **visualization of a set of co-location rules**. Figure 9.10 is the interface of CPA.

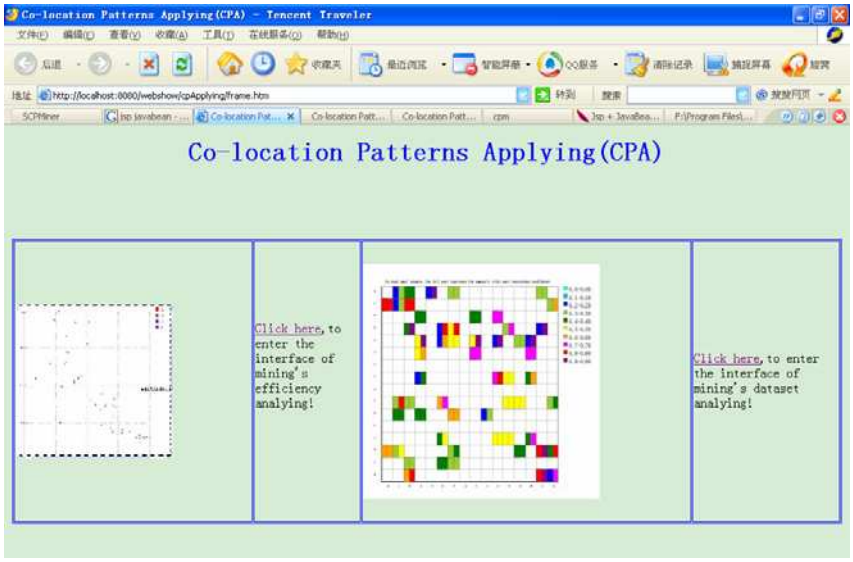


Figure 9.10 Interface of CPA

(1) **Rules' plants growth environment query**: selecting a co-location rule from mining results, you can see corresponding plants' instances distribution map, then, you can add contour line, longitude, woof or the humidity information of the area on the map. The visualization results will be significant for analyzing and researching the relationship between plants and ecological environments. Figure 9.11 is an example of rules' plants growth ecological environment (longitude and woof) query.

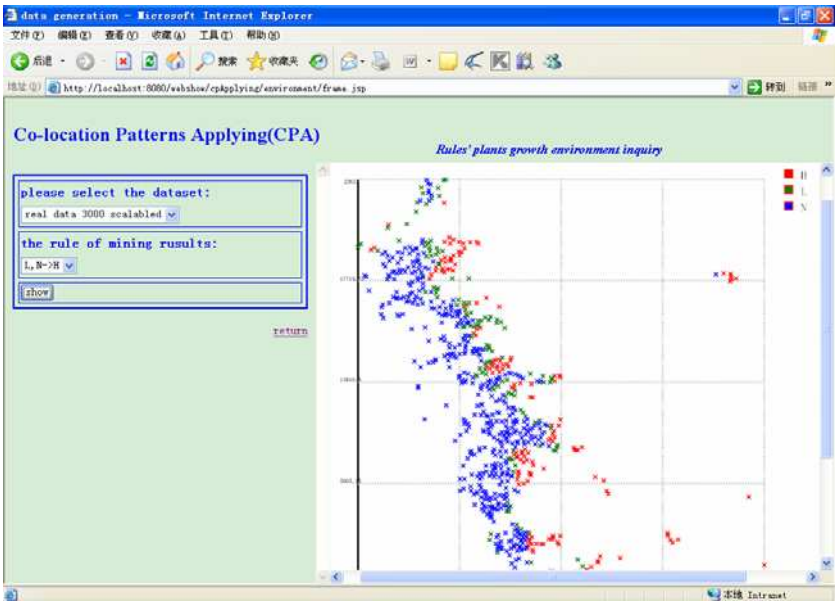
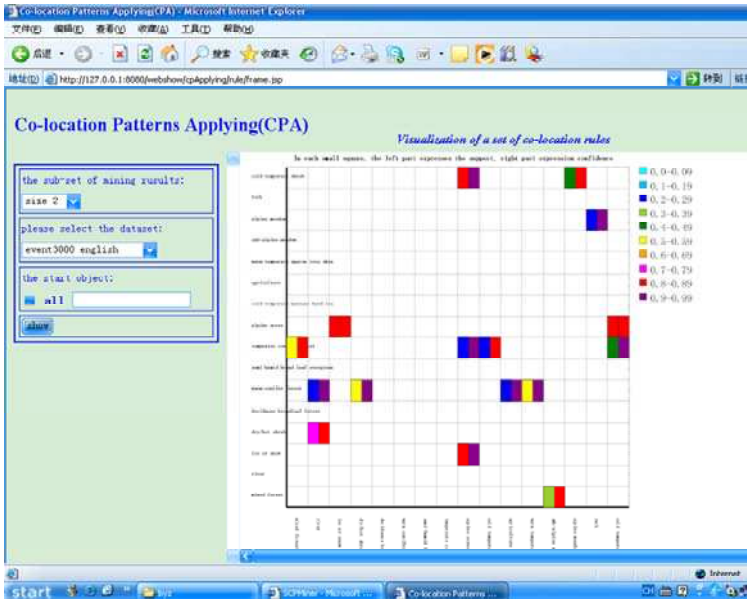


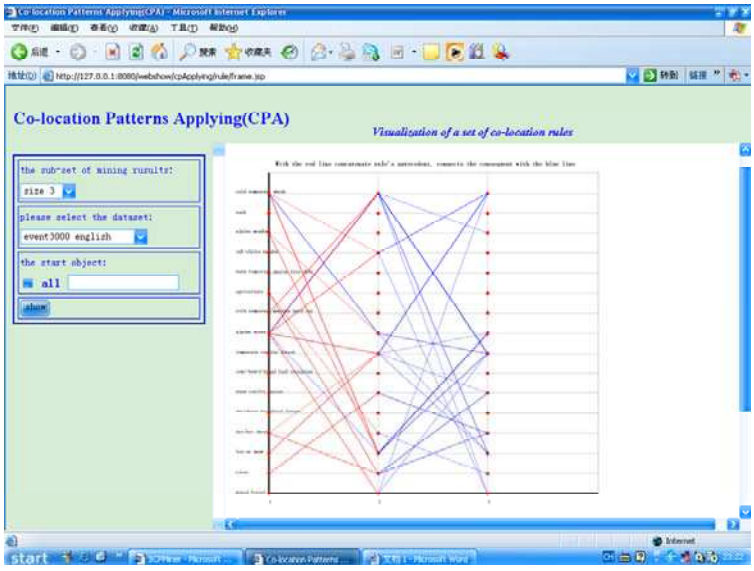
Figure 9.11 An example of rules' plants growth environment query

(2) **Visualization of a set of co-location rules:** Visualization of a sub-set of mining results is the presentation of co-location rules obtained from co-location mining in visual forms. Figure 9.12 gives visualization of size-2 and size-3 co-location rules mined from the plants distribution dataset of the "Three Parallel Rivers of Yunnan Protected Ar-

eas". Figure 9.12 (a) uses a two-dimension matrix method to describe a set of size-2 co-location rules. In this figure, the row and the column of the matrix represent the anteriority and the posterior of the rules respectively. In matrix unit, using difference colours represent the difference degree of prevalence and the conditional probability of corresponding rule. For size-3 rules and above, a parallel coordinate method is generally used. Figure 9.12 (b) is a visual result of parallel coordinate method of size-3 co-location rules mined from the plants distribution dataset of the "Three Parallel Rivers of Yunnan Protected Areas".



(a). Visualization of size-2 co-location rules



(b). Visualization of size-3 co-location rules

Figure 9.12 Visualization of co-location rules in the plants' distribution dataset of the "Three Parallel Rivers of Yunnan Protected Areas"

9.4 Summary

Data mining products, for example Intelligent Miner (an IBM data mining product), Microsoft SQL Server 2005, MineSet, Clementine (from SPSS), and et al. are fast evolving. But for spatial data mining, it is in stage of researching prototypes. In this chapter, a visual spatial co-location patterns mining prototype system (SCPMiner) is investigated.

In SCPMiner, there are not only data management function and four co-location mining methods, but also co-location mining methods analysis and co-location mining applications. This work will be a significant step towards to develop spatial data mining products.

Chapter 10

Concluding Remarks

Based on studying the evolution of spatial data mining, the thesis mainly proposed the following new techniques: the fuzzy co-location mining, **CPI-tree** (Co-location Pattern Instance Tree) which materializes spatial neighbour relationships for co-location mining, maximal co-locations mining**,** attribute-oriented induction based on attributes' generalization sequences (AOI-ags), data mining prediction, a cell-based spatial object fusion, and a fuzzy clustering based on domain knowledge. A prototype system of mining spatial co-location patterns was developed.

10.1 Contributions and Conclusions

First, the new concept of fuzzy co-location mining was proposed. Based on studying the fuzzy properties of spatial objects and spatial relationship, the motivation, basic concepts, and algorithms of discovering fuzzy co-location were expounded. In the design of algorithm, a new data structure, the binary partition tree, improving the process of fuzzy equivalence partitioning, was proposed. A prefix-based approach to partition the prevalent event set search space into subsets, where each sub-problem can be solved in main-memory, was also presented. Finally, theoretical analysis and experimental results on synthetic data sets and a real-world plant distribution dataset were presented and discussed.

Second, a new join-less method for co-location mining was proposed. A new structure called CPI-tree (Co-location Pattern Instance Tree) was introduced in the thesis. It could materialize the spatial neighbour relationships of a spatial data set, and find all the co-location table instances recursively from it. The algorithm is efficient since it does not require expensive spatial joins or instance join for identifying co-location table instances. The experimental results showed the new method outperforms the join-less method in the case of both sparse and dense datasets.

Third, an order-clique-based method for mining maximal co-location patterns was proposed. Two extended prefix-tree structures, *Neib-tree* and P_2*-tree*, which the spatial neighbour relationships between instances and the size-2 prevalence co-locations are compressed into them respectively, were introduced. A method of generating all the candidate maximal ordered prevalence co-locations from the P_2*-tree* was presented. An algorithm of inspecting all table instances from the *Neib-tree* was given. In this algorithm, the table instances do not need be stored after computing the *Pi* value of corresponding

co-location, which dramatically reduces the executive time and space of mining maximal co-locations. A performance study has been conducted to compare the performance of the *order-clique-based* method with three representative co-location mining methods, the *full-join,* the *join-less* and the *CPI-tree*. The study shows that *order-clique-based* method is much faster than *full-join, join-less* and the *CPI-tree,* especially when the spatial data-set is dense (containing many table instances) and/or when the prevalence co-locations are long.

Fourth, by introducing the concept of the attributes' generalization sequence, the attribute thresholds and the tuple thresholds were unified, and a reasonable approach of AOI—AOI-ags (Attribute-Oriented Induction based on Attributes' Generalization Sequences), which expands the traditional AOI, was proposed. Some technologies, for example, partitions, equivalence partition trees, prune optimization strategies and interestingness, were used to improve the efficiency of the algorithm. It was shown that the AOI-ags algorithm has its advantages.

Fifth, based on the semantic proximity, a mining prediction method to evaluate the fuzzy association degree was given. Inverse document frequency (IDF) weight function has been adopted in this investigation to measure the weights of conditional attributes in order to superpose the fuzzy association degrees. To implement the method, the "growing window" and the proximity computation pruning were introduced to reduce both I/O and CPU costs in computing the fuzzy semantic proximity between time-series. Extensive experiments on real datasets were conducted, and the results showed that the mining prediction approach is reasonable and effective.

Sixth, a cell-based spatial object fusion method was proposed. The main contribution of this work is in showing that corresponding objects can be effectively found without distance between objects.

Seventh, In order to get the tuples' semantic proximity matrix, domain knowledge was used in fuzzy clustering. Two clustering methods: Natural Method and Graph-Based Method, both of which is controlled by a threshold, and the threshold is confirmed by polynomial regression, were proposed.

Finally, a prototype system of spatial co-location patterns mining was described. In this system, experimental data and real-world data were managed. A user-friendly interface of mining was developed. The mining results could be checked conveniently. This work is a significant step for investigating of spatial patterns mining products.

10.2 Forecasting Perspectives

Spatial data mining is a rising field in spatial information science. Some achievements in this area have been gained, many challenges, however, still remain. Where, mainly including the following directions: multirelational and multidatabase spatial data mining, uncertain spatial data mining, multilevel spatial data mining, parallel spatial data mining, the investigation of new methods and efficient algorithms, the design of spatial data mining languages, and the effective application of spatial data mining techniques. More investigations are required, especially toward the integration of spatial data mining methods with object-oriented spatiotemporal databases, spatial index, spatial reasoning, spatial data warehouses, and GIS.

Spatial data mining, which could change the limited spatial data into unlimited spatial knowledge, has extensive forecasting perspectives and potential general benefits. Today, as spatial information increasing and development of software and hardware techniques, spatial data mining has infiltrated into GIS, information fusion, preprocessing of remote sensing data, medical imaging processing, navigation, robot, et al. It will become an actual that using the discoverable spatial knowledge accelerates researching on automatization and intelligentize of these subjects.

References

Agarwal, R. and Srikant, R. (1994) Fast Algorithms for Mining Association Rules, *In*: Proc. 1994 Int. Conf. Very Large Data bases (VLDB'94), Santiago, Chile, Sept. 1994, pp. 487-499

Agrawal, R., Gehrke, J., Gunopulos, D. and Raghavan, P. (1998) Automatic subspace clustering of high dimensional data for data mining applications, *In*: Proceedings of the ACM SIGMOD Conference on Management of Data, Seattle, WA, 1998, pp. 94–105

Alsabti, K., Ranka, S. And Singh, V. (1998) CLOUDS: A Decision Tree Classifier for Large Databases. *In:* Proceedings of the 4th Intl. Conf. on Knowledge Discovery and Data Mining, New York, August 1998, pp. 2-8.

Alsuwaiyel, M. H. (2004) Algorithms Design Techniques and Analysis. Publishing House of Electronics Industry, Beijing, 2004

Amstader, B.L. (1979) Reliability Methematics, McGraw-Hill, New York, 1979

Beeri, C., Kanza, Y., Safra, E. and Sagiv, Y. (2004) Object Fusion in Geographic Information Systems, *In:* proceedings of the 30th VLDB Conference, Toronto, Canada, 2004, pp. 816-827.

Bruns, T. and Egenhofer, M. J. (1996) Similarity of spatial scenes, *In:* Proceedings of the 7th International Symposium on Spatial Data Handling, Delft (Netherlands), 1996, pp. 31-42

Buckles, B.P. and Petry, F.E. (1982) 'A fuzzy representation of data for relational databases' *Fuzzy Sets and Systems*, 7, (3) pp. 213-226

Cai, Y., Cercone, N. and Han, J. (1991) Attribute-Oriented Induction in Relational Databases, Piatetsky-Shapiro, G. and Frawley, W.J. eds, *Knowledge Discovery in Databases*, AAAI/MIT Press, Menlo Park, Calif., pp. 213-228

Carter, C. L. and Hamilton, H. J. (1998) 'Efficient attributed-oriented generalization for knowledge discovery from large databases' *IEEE Trans. on Knowledge and Data Eng.*, 10, (2) pp. 193-208

Celik, M., Kang, J. M. and Shekhar, S. (2007) Zonal Co-location Pattern Discovery with Dynamic Parameters, In: Proc. of the Seventh IEEE International Conference on Data Mining (ICDM'07), Omaha, United States, 2007, pp.433-438

Chou, Y. (1997) *Exploring Spatial Analysis in Geographic Information System*. Onward Press

Cressie, N.A.C. (1991) *Statistics for Spatial Data*. Wiley and Sons

Ester, M., Kriegel, H. -P., Sander, J., and Xu, X. (1996) A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases, In: Proc.1996 Int. Conf. Knowledge Discovery and Data Mining (KDD'96). pp: 226-231, Portland, OR, Aug. 1996

Ester, M., Kriegel, H.-P., and Sander, J. (1997) Spatial Data Mining: A Database Approach, *In:* Proc. 5th Int. Symposium on Large Spatial Databases (*SSD'97*), Berlin, Germany, 1997, pp. 47-66

Ester, M., Kriegel, H.-P., Sander, J. and Xu, X. (1997) Density-connected sets and their application for trend detection in spatial databases, *In:* Proc. 1997 Int. Conf. Knowledge Discovery and Data Mining (*KDD' 97*), Newport Beach, CA, Aug.

1997, pp. 10-15

Ester, M., Frommelt, A., Kriegel, H.-P., and Sander, J. (1998) Algorithms for Characterization and Trend Detection in Spatial Databases, *In:* Proc. 4th Int. Conf. on Knowledge Discovery and Data Mining (*KDD'98*), New York, NY, 1998, pp. 44-50

Ester, M., Kriegel, H.-P., and Sander, J. (1999) Knowledge Discovery in Spatial Databases, *In:* invited paper at 23rd German Conf. on Artificial Intelligence *(KI '99*), Bonn, Germany, *Lecture Notes in Computer Science*, Vol. 1701, pp. 61-74.

Estivill-Castro, V. and Murray, A. (1998) Discovering Associations in Spatial Data—An Efficient Medoid Based Approach, *In:* Proc. Second Pacific-Asia Conf. Knowledge Discovery and Data Mining, PAKDD-98, Springer-Verlag, Berlin. pp.110-121.

Estivill-Castro, V. and Lee, L. (2001) Data Mining Techniques for Autonomous Exploration of Large Volumes of Geo-Referenced Crime Data, *In:* Proc. Sixth Int'l Conf. Geocomputation, 2001.

Fonseca, F. T. and Egenhofer, M. J. (1999) Ontology-driven geographic information systems, *In:* Proceedings of the 7th ACM International Symposium on Advances in Geographic Information Systems, Kansas City (Missouri, US), 1999, pp. 14-19

Fonseca, F. T., Egenhofer, M. J. and Agouris, P. (2002) 'Using ontology for integrated geographic information systems' *Transactions in GIS,* 6, (3) pp. 231-257

Graefe, G. (1994) Sort-Merge-Join: An Idea Whose Time Has (h) Passed? *In:* Proc. IEEE Conf. Data Eng., 1994, pp. 406-417

Guo, L. (2004) Geological Dividing of Forests at Three Parallel Rivers of Yunnan Protected Areas, Journal of West China Forestry Science, 33 (2) (2004) 10-15

Han, J., Cai, Y. and Cercone, N. (1993) Data-driven discovery of quantitative rules in relational databases. IEEE Trans. Knowledge and Data Engineering, 5:29-4-, 1993

Han, J. (1994) Towards Efficient induction mechanisms in database systems, *Theoretical Computing Science*, 1994, 133, pp. 361-385

Han, J. and Fu, Y. (1996) Exploration of the power of attribute-oriented induction in data mining. In U. M. Fayyad, et al, editors, Advances in Knowledge Discovery and Data Mining, pp. 399-421, AAAI/MIT Press, 1996

Han, J., Koperski, K. and Stefanovic, N. (1997) GeoMiner: A system prototype for spatial data mining, *In:* Proc. 1997 ACM-SIGMOD Int. Conf. Management of Data (*SIGMOD'97*), Tucson, AZ, May 1997, pp.553-556

Han, J. and Kamber, M. (2006) *Data mining: concepts and techniques* (Second Edition), China Machine Press, Beijing

Hastie, T., Tibshirani, R. and Friedman, J. H. (2001) The Elements of Statistical Learning: Data mining, Inference, Prediction, Springer, New York, 2001

He, X. (1989) 'Semantic distance and fuzzy user's view in fuzzy database' *Chinese J. Comput*, 10, (1989) pp. 757-764

Huo, Z. (1989) *Fuzzy Mathematics and its Applications*, Tianjin Science and Technology Press, Tianjin, China

Huang, Y., Pei, J. and Xiong, H. (2006) 'Mining Co-location Patterns with Rare Events from Spatial Data Sets', Geoinformatica (2006) 10:239-260.

Huang, Y., Shekhar, S. and Xiong, H. (2004) 'Discovering Colocation Patterns from Spatial Data Sets: A General Approach' *IEEE Transactions on Knowledge and Data Engineering*, 16, (12) pp. 1472-1485

Huang, Y. and Zhang, P. (2006) On the Relationships between Clustering and Spatial Co-location Pattern Mining, *In:* Proc. of the 18th IEEE Int. Conf. on Tools with Artificial Intelligence (ICTAI 06), Washington D.C., Nov. 2006, pp: 513 – 522.

Kaufman, L. and Rousseeuw, P. J. (1990) Finding Group in Data: An introduction to cluster analysis, John Wiley & Sons, 1990.

Knorr, E. M. and Ng, R. T. (1997) Extraction of spatial proximity patterns by concept generalization, *In:* Conf. on spatial information theory (*COSIT*), 1997, pp. 15-33

Kohavi, R. and Sahanu, M. (1996) Error-based and Entropy-based Discretization of Continuous Features, In: Proc. of the 2nd International Conference on Knowledge Discovery and Data Mining, Portland, Oregon: AAAI Press, 1996-08: 114-119

Koperski, K. and Han, J. (1995) Discovery of Spatial Association Rules in Geographic Information Databases, *In:* Proc. 1995 Int. Symp. Large Spatial Databases (*SSD'95*), Portland, ME, Aug. 1995, pp. 47-66

Koperski, K., Han, J. andStefanovic, N. (1998) An efficient two-step method for classification of spatial data, *In:* Proc. 8th Symp. Spatial Data Handling, Vancouver, Canada, 1998, pp. 45-55

Kriegel, H.-P., Pfeifle, M. and Schönauer, S. (2004) 'Similarity Search in Biological and Engineering Databases' *IEEE Data Engineering Bulletin*, 27, (4) pp. 37-44

Larsen, H. L. and Yager, R. R. (1990) 'Efficient computing of transitive closures' *Fuzzy Sets and Systems*, 38, (1) pp. 81-90

Lee, A. J. T., Hong, R., Ko, W., Tsao, W., and Lin, H. (2007) 'Mining spatial association rules in image databases' Information Sciences, 177 (2007) 1593-1608.

Li, D., Wang, S. and Li, D. (2006) *Spatial Data Mining Theories and Applications*, Science Press, Beijing, 2006

Lin, C. and Chen, M. (2005) 'Combining Partitional and Hierarchical Algorithms for Robust and Efficient Data Clustering with Cohesion Self-Merging' IEEE Trans. Knowl. Data Eng. 17(2): 145-159 (2005)

Liu, W. (1993) 'The fuzzy functional dependency on the bases of the semantic distance' Fuzzy Sets and Systems 59 (2) (1993) 173-179

Liu, W. and Song, N. (2001) 'The fuzzy association degree in semantic data models' *Fuzzy Sets and Systems*, 117, (2001) pp. 203-208

Liu, W. and Tian, W. (2001) *Data model*, Science Press, Beijing, 2001.

Lu, J., Wang, L., Li, Y. (2007) A Fuzzy Clustering Method Based on Domain Knowledge. In: Proceedings of the Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD2007), Qingdao, China, July30-Aug.1, 2007, pp.297-302.

Lu, W., Han, J. and Ooi, B.C. (1993) Discovery of general knowledge in large spatial databases, *In:* Proc. Far East Workshop on Geographic Information Systems, Singapore, June 1993, pp. 275-289

MacQueen, J. (1967) Some methods for classification and analysis of multivariate observations. In: Proc. 5th Berkeley Symp. Math. Statist. Prob., 1: 281-297, 1967

Minami, M. (2000) Using ArcMap, Environmental Systems Research Institute, Inc., 2000.

Morimoto, Y. (2001) Mining Frequent Neighbouring Class Sets in Spatial Databases, *In:* Proc. ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining, 2001. pp. 353-358

Ng, R. and Han, J. (1994) Efficient and effective clustering method for spatial data mining, *In:* Proc. 1994 Int. Conf. Very Large Data Bases (*VLDB'94*), Santiago, Chile, Sept. 1994, pp. 144-155

Papakonstantinou, Y., Abiteboul, S. and Garcia-Molina, H. (1996) Object Fusion in Mediator Systems, *In:* proceedings of the 22nd VLDB Conference, Mumbai (Bombay), India (1996), pp. 413-424

Samal, A., Seth, S. and Cueto, K. (2004) 'A feature based approach to conflation of geospatial sources' *International Journal of Geographical Information Science*, 18, (00) (2004), pp. 1-31

Schwartz, D.G. (1989) 'Fuzzy inference in a formal theory of semantic equivalence' *Fuzzy Sets and Systems*, 31, (2) pp. 205-216

Shekhar, S. and Chawla, S. (2003) *Spatial Databases: A Tour*, Prentice Hall, 2003

Shekhar, S. and Huang, Y. (2001) Co-location Rules Mining: A Summary of Results, *In:* Proc. of International Symposium on Spatio and Temporal Database (SSTD), 2001

Shenoi, S. and Melton, A. (1989) 'Proximity relations in the fuzzy relational database model' *Fuzzy Sets and Systems,* 31, (3) pp. 285-296

Shi, Y., Song, Y. and Zhang, A. (2003) A Shrinking-Based Approach for Multi-dimensional Data Analysis, *In:* proceedings of the 29th International Conference on Very Large Data Bases (VLDB03), Berlin, Germany, 2003, pp.440-451

Tarjan, R. E. and Leeuwen, T. (1984) 'Worst-case analysis of set union algorithms' *J. ACM,* 31, (2) pp. 245-281

Tay, F. E. F. and Shen, L. (2002) 'A Modified Chi2 Algorithm for Discretization' , IEEE Transactions on Knowledge and Data Engineering, 2002, 14(3): 666-670

Uitermark, H., Oosterom, P. V., Mars, N. and Molenaar, M. (1999) Ontology-based geographic data set integration, *In:* Proceedings of Workshop on Spatio-Temporal Database Management, Edinburgh (Scotland), 1999, pp. 60-79

Wang, W., Yang, J. and Muntz, R. (1997) STING: A Statistical Information Grid Approach to Spatial Data Mining, *In:* proceedings of the 23rd VLDB Conference, Athens, Greece, 1997, pp. 186-195

Wang, L. (2000) A method of the abstract generalization on the bases of the semantic proximity, *CHINESE J. COMPUTERS*, 23, (10) pp. 1114-1121

Wang, L. and Chen, H. (2005) Record Reduction Based on Attribute Oriented Generalization, *In:* Proceedings of the Fourth International Conference on Machine Learning and Cybernetics (ICMLC05), Guangzhou, China, 2005, pp. 1693-1700

Wang, L., Xie, K., Chen, T. and Ma, X. (2005) 'Efficient discovery of multilevel spatial association rule using partition' *Information and Software Technology (IST)*, 47, (13) pp. 829-840

Wang, L. and Li, H. (2006) A Cell-Based Spatial Object Fusion Method, *In:* Proceedings of the International Conference on Complex Systems and Applications (ICCSA06), Huhhot, China, June 16-18, 2006, pp.106-110

Wang, L., Lu, J., Yip, J. (2007) AOG-ags Algorithms and Applications, ADMA 2007, LNAI

4632, pp. 323-334, 2007.8

Wang, L., Lu, J., Yip, J. (2007) An Effective Approach to Predicting Plant Species in an Ecological Environment, *In:* Proceedings of the 2007 international Conference on Information and Knowledge Engineering (IKE’07), June 25-28, 2007, Las Vegas Nevada, USA, pp. 245-250

Wang, L., Yang, A., Zhang, H. (2007) Data Mining Prediction of Shovel Cable Service Lifespan. SNPD 2007（Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing）, Qingdao, China, July 30-Aug.1, 2007, pp.233-238

Xiong, H., Shekhar, S., Huang, Y., Kumar, V., Ma, X. and Yoo, J. S. (2004) A Framework for Discovering Co-location Patterns in Data Sets with Extended Spatial Objects, *In:* Proc. 2004 SIAM International Conference on Data Mining (*SDM*), 2004, pp. 1-12

Yoo, J. S. and Shekhar, S. (2004) A partial Join Approach for Mining Co-location Patterns, *In:* Proc. of the 12th annual ACM international workshop on Geographic information systems, 2004, pp. 241-249

Yoo, J. S., Shekhar, S. and Celik, M. (2005) A Join-Less Approach for Co-Location Pattern Mining: A Summary of Results, *In:* Proc. of the 5th IEEE Int. Conf. on Data Mining, ICDM 2005, Houston, November 2005, pp. 813-816.

Zadeh, L.A. (1965) ‘Fuzzy sets’ *Inform. and Control*, 8, (3) pp. 338-353

Zaiane, O. R. et al., (1998) Multimedia-miner: a system prototype for multimedia data mining, In: Proceedings of 1998 CMSIGMOD Conference on Manegement of Data, (system demo), Seattle, Washington, June

Ziarko, W. (1991) The discovery, analysis and representation of data dependency in databases, *In:* G.P. Shapiro (Ed.), Knowledge Discovery in Databases, Benjamin/Cummings, Menlo Park, CA, 1991, pp. 213-228

Zaki, M. J. (2000) ‘Scalable Algorithms for Association Mining’ *IEEE Transactions on Knowledge and Data Engineering*, 12, (3) pp. 372-390

Zhang, M., Wang, D., and Yu, G. (2004) ‘A Text Clustering Method Based on Auto Selected Threshold’, Journal of Computer Research and Development, 2004, 41(10): 1748-1753

Zhang, T., Ramakrishnan, R., and Livny, M. (1996) BIRCH: an efficient data clustering method for very large databases, *In:* Proceedings of 1996 *ACM-SIGMOD* Int. Conf. Management of Data (SIGMOD’96), Pages 103-114, Montreal, Canada, June 1996

Zhou, X., Truffet, D. and Han, J. (1999) Efficient polygon amalgamation methods for spatial OLAP and spatial data mining, *In:* Proc. 1999 Int. Symp. Large Spatial Databases (*SSD’99*), Hong Kong, China, July 1999, pp. 167-187

Appendix 1

The Partial Codes of SCPMiner

```
//The following codes is for generating data of co-location data management function:
Left.jsp
<%@page contentType="text/html;charset=gb2312"%>
<%@ page import="java.sql.*" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>no title file</title>
<style type="text/css">
<!--
.style11 {font-size: 16px; color: #0000FF; font-weight: bold; }
-->
</style>
</head>

<body  bgcolor="#D6ECD5">
<form name="form1" method="post" action="show.jsp" target="main" onsubmit="return check-
Login();">
  <table width="100%"  border="2" cellspacing="5" bordercolor="#0000FF">
    <tr>
      <td><table width="100%"  border="0" cellspacing="0">
        <tr>
          <td height="25"><span class="style11">feature:</span> </td>
        </tr>
        <tr>
          <td height="25"><span class="style11">from:
              <select name="start"; id="start">
 <jsp:useBean id="RegisterBean" scope="page" class="colocation.Opendb" />
<%
  String sql="select * from type";
  ResultSet rs=RegisterBean.executeQuery(sql);
  while (rs.next()){
  String s1,s2;
  s1=rs.getString(1);
  s2=rs.getString(2);

  %>

                    <option value="<%=s1%>"><%=s2%></option>
  <%


  }
    rs.close();
%>
            </select>
          </span></td>
        </tr>
        <tr>
```

```
        <td height="25"><span class="style11">to:
            <select name="end"; id="end">

<%
  String sql2="select * from type";
  ResultSet rs2=RegisterBean.executeQuery(sql2);
  while (rs2.next()){
   String s1,s2;
  s1=rs2.getString(1);
  s2=rs2.getString(2);

  %>

                    <option value="<%=s1%>"><%=s2%></option>
  <%

  }
    rs2.close();
%>
            </select>
          </span></td>
        </tr>
      </table></td>
    </tr>
    <tr>
      <td><table width="100%"  border="0" cellspacing="0">
        <tr>
          <td height="25"><span class="style11">max number of instance:</span></td>
        </tr>
        <tr>
          <td height="25"><span class="style11">
          <input name="maxCount" type="text" id="maxCount" value="100" size="10" max-
length="10">
          </span></td>
        </tr>
      </table></td>
    </tr>
    <tr>
      <td><table width="100%"  border="0" cellspacing="0">
        <tr>
          <td height="25"><span class="style11">scop:</span></td>
        </tr>
        <tr>
          <td height="25"><span class="style11">width:</span><span class="style11">
            <input name="width" type="text" id="width" value="1000" size="10" maxlength="10">
          </span></td>
        </tr>
        <tr>
          <td height="25"><span class="style11">height:</span><span class="style11">
          <input name="height" type="text" id="height" value="1000" size="10" maxlength="10">
          </span></td>
        </tr>
      </table></td>
    </tr>
    <tr>
      <td><table width="100%"  border="0" cellspacing="0">
        <tr>
          <td height="25"><span class="style11">move:</span></td>
        </tr>
        <tr>
```

```
        <td height="25"><span class="style11">x:</span><span class="style11">
          <input name="startX" type="text" id="startX" value="0" size="10" maxlength="10">
        </span></td>
       </tr>
       <tr>
        <td height="25"><span class="style11">y:</span><span class="style11">
          <input name="startY" type="text" id="startY" value="0" size="10" maxlength="10">
        </span></td>
       </tr>
     </table></td>
   </tr>
   <tr>
     <td><table width="100%"  border="0" cellspacing="0">

       <tr>
        <td height="25"><span class="style11">file name: </span></td>
       </tr>
       <tr>
        <td height="25"><input name="filename" type="text" id="filename" value="eventlx"
size="30" maxlength="30"></td>
       </tr>
     </table></td>
   </tr>
   <tr>
     <td><input type="submit" name="Submit" value="gen data">
     <input type="reset" name="Submit2" value="result"></td>
   </tr>

  </table>
  </tr>
  </table>
</form>
</body>
</html>

Show.jsp
<%@page contentType="text/html;charset=gbk"%>
<jsp:useBean id="gendata" scope="session" class="gendata.genData" />
<jsp:useBean id="svg" scope="session" class="SVG_ScatterGraphExample.SvgScatterSjk1" />

<%
//  out.print("<h3>runing.....</h3>");
  request.setCharacterEncoding("GBK");
%>

<%!
String start=null;
String end=null;
String maxCount=null;
String width=null;
String height=null;
String startX=null;
String startY=null;
String filename=null;
%>

<%
start=request.getParameter("start");
end=request.getParameter("end");
maxCount=request.getParameter("maxCount");
```

```
width=request.getParameter("width");
height=request.getParameter("height");
startX=request.getParameter("startX");
startY=request.getParameter("startY");
filename=request.getParameter("filename").trim();
%>
<html>
   <body>
  <%


  gendata.mainProg(Integer.parseInt(start),Integer.parseInt(end),Integer.parseInt(maxCount),
  Integer.parseInt(width),Integer.parseInt(height),
  Integer.parseInt(startX),Integer.parseInt(startY),filename);

  String ss=filename+".svg";
   svg.mainProg(filename,filename);

   //out.print("<h3>run over!</h3>");
   %>
    <embed name="svg" type="image/svg+xml" src="../brow/<%=ss%>" width="700"
height="700">
  </body>
</html>
genData.java
package gendata;
import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Random;


public class genData {
  String filename;
  String[] type;

   /**
    * Read data from the file, and then put the events and instances into a two-dimensional array
respectively
    * @throws java.io.IOException  the exceptions thrown by the file processing
    */
   public void readData(Statement stmt,int start,int end) throws IOException,SQLException {
     //start: The number of the start type
     //end: The number of the end type
     int k=end-start+1;
     this.type=new String[k];
       String typeTemp="";
       int idTemp=0;

       String sqlStr="select id,type from useType order by 1,2";
     ResultSet rs = stmt.executeQuery(sqlStr);
    int i=0;
    while (rs.next()) {
      idTemp=Integer.valueOf(rs.getString(1));
      typeTemp=rs.getString(2).trim();

      System.out.println(typeTemp+idTemp);
```

```
    if (idTemp>=start && idTemp<=end){
       type[i]=typeTemp;
      ++i;
      }

  }
 }

 public void randomSj(String type,int maxCount,int width,int height,
    int startX,int startY,Statement stmt) throws SQLException{
  //maxCount: The maximal count of each type
  //width:
  //height:
  //startX:
  //startY:
      Random ranInt=new Random();
      String s3;
      int ix,iy,j,gs;
      gs=ranInt.nextInt(maxCount);
      while(gs==0){
        gs=ranInt.nextInt(maxCount);
      }
    for(int i=0;i<gs;i++){
      j=i+1;
      s3="insert into "+filename+"(type,id,x,y) values('"+type+"',"+j+",";
      ix=ranInt.nextInt(width)+startX;
      iy=ranInt.nextInt(height)+startY;
      s3=s3+ix+","+iy+" )";

    //System.out.println("s3="+s3);
    stmt.executeUpdate(s3);
    //Add some tuples into the instance-table
    }
 }


 public void genTable(Statement stmt) throws SQLException{
   //stmt.executeUpdate("CREATE TABLE col_link (sitename varchar (20) NULL ,siteurl varchar
(50) NULL) ");
   String createP="create table "+filename+"( type char(10),id decimal(9),x decimal(9),y deci-
mal(9))";
   //System.out.println(createP);
    ResultSet rt=stmt.executeQuery("select count(*) as n  from sysobjects where name
='"+filename+"'");
   if (rt.next()) {
     int num=Integer.valueOf(rt.getString("n"));
     if (num>0){
       stmt.executeUpdate("drop table "+filename);
     }
   }

   stmt.executeUpdate(createP);

 }

 public void mainProg(int start,int end,int maxCount,
     int width,int height,int startX,int startY,
     String filename) throws IOException,SQLException{
```

```
    //The number of points, the dimensionality of points generated and the bound of the points
  gererated
     this.filename=filename;

   String url = "jdbc:odbc:sqlFullJoin";
   Connection con;
   Statement stmt;
   try {
     Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
   }
   catch(java.lang.ClassNotFoundException e) {
     System.err.print("ClassNotFoundException: ");
     System.err.println(e.getMessage());
   }
   try
   {
     con = DriverManager.getConnection(url, "sa", "75");
     stmt = con.createStatement();
     genTable(stmt);
     readData(stmt,start,end);
     for(int i=0;i<type.length;i++){
       randomSj(type[i],maxCount,width,height,
           startX,startY,stmt);
     }
     stmt.close();
     con.close();
   }
   catch(SQLException ex) {
     System.err.println("SQLException: " + ex.getMessage());
   }
  }


  public static void main(String args[]) throws IOException,SQLException{
    int start=1;
    int end=4;
    int maxCount=10;
    int width=100;
    int height=20;
    int startX=20;
    int startY=20;
    String filename="eventlx";
       genData lx=new genData();
      // Integer.parseInt("1");
       lx.mainProg(start,end,maxCount,width,height,startX,startY,filename);
       System.out.println("Running finished!");

  }

}


**************************************************************************
FullJoin.java  //Join-based algorithm
package SFullSjk;

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
```

```
import java.util.*;
import java.sql.*;

public class FullJoin {
  private LinkedHashMap<String,List<Instance>> eventmap=new LinkedHash-
Map<String,List<Instance>>() ;
  private LinkedHashMap<Model,List<ModelInstance>> distancePair;
  private LinkedHashMap<Model,List<ModelInstance>> lc; //The set lc of candidate prevalence
co-location patterns
  private LinkedHashMap<Model,List<ModelInstance>> lc2; // The set lc2 of candidate preva-
lence co-location patterns
  private LinkedHashMap<Model,Double> lk;// The set lk of the size-k candidate prevalence co-
location patterns
  private LinkedHashMap<Model,Double> lk2;// The set lk2 of the size-k+1 candidate prevalence
co-location patterns
  private String filename;
  // lk lc ->lk2,lc2

  double distance;//The distance D
  private double minSup;//The min_prev
  private String eTime;

  int k;
  public void read(Statement stmt,String filename) throws IOException,SQLException {
    eventmap=new LinkedHashMap<String,List<Instance>>();
    String eventType;
    String eventTypeTemp="";
    int instanceNumber=0;
    double x=0;
    double y=0;
    List<Instance> listInstance;
    listInstance=new ArrayList<Instance>();
    String sqlStr="select * from "+filename.trim()+" order by 1,2";
    ResultSet rs = stmt.executeQuery(sqlStr);
    //  Return a set of results
    //System.out.println("the event originality data:");
    //System.out.println("type "+" "+"id"+"x"+"y");

    while (rs.next()) {
      eventType=rs.getString(1).trim();
      String bl=rs.getString(2);
      instanceNumber=Integer.valueOf(bl);
      x=Double.valueOf( rs.getString(3));
      y=Double.valueOf(rs.getString(4));
      /*
       * eventType=rs.getString("event_type").trim();
      String bl=rs.getString("event_id");
      instanceNumber=Integer.valueOf(bl);
      x=Double.valueOf( rs.getString("x"));
      y=Double.valueOf(rs.getString("y"));
       */

      Instance instance1=new Instance(instanceNumber,x,y);
      //System.out.println(eventType + " " + instanceNumber+" "+x+" "+y);
      if (eventmap.keySet().contains(eventType)==false){
        listInstance=new ArrayList<Instance>();
        eventmap.put(eventType, listInstance);
      }
      eventmap.get(eventType).add(instance1);
```

```
    }//end file

  }//end FileReadIn

  public LinkedHashMap<Model,List<ModelInstance>>
   caculateDistance(double disParameter){

       LinkedHashMap<Model,List<ModelInstance>> distancePair=new LinkedHash-
Map<Model,List<ModelInstance>>();
       Model joinEvent;
       List<ModelInstance> listModelInstance;

       List<String> eventList=new ArrayList<String>();
       String eventOne,eventTwo;

       List<Instance> instanceListOne,instanceListTwo;
       eventList.addAll(eventmap.keySet());
       //System.out.println("the eventlist is:"+eventList);
       for(int i=0;i<eventList.size()-1;i++){
          eventOne=eventList.get(i);
          instanceListOne=eventmap.get(eventOne);

          //The following is to print event1 and its instances
          //System.out.println("The following is to print event1 and its instances ");
          //System.out.println("The instances are:"+eventOne);
          /*
          System.out.println("the one instance list is:");
          for(int k=0;k<instanceListOne.size();k++)
          {
           instanceListOne.get(k).printInstance();
          }

          */
          for(int j=i+1;j<eventList.size();j++){
           listModelInstance=new ArrayList<ModelInstance>();
             eventTwo=eventList.get(j);
             instanceListTwo=eventmap.get(eventTwo);


             if(distance(listModelInstance,
                instanceListOne,
                instanceListTwo,
                disParameter)
                ){
                joinEvent=new Model();
                joinEvent.list.add(eventOne);
                joinEvent.list.add(eventTwo);
                distancePair.put(joinEvent,listModelInstance);
                //System.out.println("***************");
                /*
                joinEvent.printList();
                for(int k=0;k<listModelInstance.size();k++)
                  System.out.println(listModelInstance.get(k).toString());
                */

             }

          }
       }
      return distancePair;
```

```
    }

//computing the distances
 public boolean distance(List<ModelInstance> listModelInstance,
     List<Instance> instanceList1,
     List<Instance> instanceList2,
     double disParameter){

   Instance instance1,instance2;
   double result;
   ModelInstance modelInstance;
   for(int i=0;i<instanceList1.size();i++){
     instance1=instanceList1.get(i);
     for(int j=0;j<instanceList2.size();j++){
       instance2=instanceList2.get(j);
       result=Math.sqrt(
       (instance1.getX()-instance2.getX())*
       (instance1.getX()-instance2.getX())+
       (instance1.getY()-instance2.getY())*
       (instance1.getY()-instance2.getY()) );

       if (result<=disParameter){
         //System.out.println(result);
         modelInstance=new ModelInstance();
         modelInstance.add(instance1.getNumber());
         modelInstance.add(instance2.getNumber());
         listModelInstance.add(modelInstance);

       }

     }

   }
   if(listModelInstance.size()>0)
     return true;
   return false;

 }


//printing the originality data
public void printEvent(){
  Set s1=eventmap.keySet();
  Iterator<String> itEvent=s1.iterator();
  while(itEvent.hasNext()){
    String event=itEvent.next();
    System.out.println("the key is:"+event);
    List<Instance> listinstance1=eventmap.get(event);
    System.out.println("listinstance.size():"+listinstance1.size());
    Iterator<Instance> itInstance=listinstance1.iterator();
    while(itInstance.hasNext()){
      System.out.println(itInstance.next().toString());
    }

  }

}

public void printPK(LinkedHashMap<Model,List<ModelInstance>> pk,int i,String message){
  System.out.println(message+i+" the start of printing");
```

```
    Set s1=pk.keySet();
    Iterator<Model> it=s1.iterator();
    while(it.hasNext()){
      Model model=it.next();
      List<ModelInstance> listinstance1=pk.get(model);
      System.out.print(model);
      System.out.println(" length="+listinstance1.size());
      Iterator<ModelInstance> itInstance=listinstance1.iterator();
      while(itInstance.hasNext()){
        System.out.println(itInstance.next().toString());
      }

    }
    System.out.println(message+i+"the end of printing");

  }

  //Printing the Pi values of the co-locations
  public void printSup(LinkedHashMap<Model,Double> lk,int i){
    System.out.println("The Pi value"+i+"is:");
    System.out.println(lk.keySet().toString());
    System.out.println(lk.values().toString());
  }

  //Computing the Pi values of the co-locations
  public  LinkedHashMap<Model,Double> calculateSup
  ( LinkedHashMap<Model,List<ModelInstance>> lc,int k,Statement stmt)
  throws SQLException {
    lk=new LinkedHashMap<Model,Double>();
    Set s2=lc.keySet();
    List listDistinct;
    Integer sz;
    double supTemp=1.0;
    double supNow=1.0;
    double everyCount=1.0;
    double everyOCount=1.0;
    String eventType;

    //Generating an instance-table
    //stmt.executeUpdate("CREATE TABLE col_link (sitename varchar (20) NULL ,siteurl varchar
(50) NULL) ");
    String createP="create table p"+k+"(INum decimal(10),";
    for(int i=1;i<=k;i++){
      createP=createP+"type"+i+" char(10),";
    }
      ResultSet   rt=stmt.executeQuery("select count(*) as n " +
          "from sysobjects where name  ='p"+k+"'");
    if (rt.next()) {
      int num=Integer.valueOf(rt.getString("n"));
      if (num>0){
        stmt.executeUpdate("drop table p"+k);
      }
    }

    createP=createP+"pi decimal(10,2))";
    //System.out.println(createP);
    stmt.executeUpdate(createP);

    Iterator<Model> it2=s2.iterator();
    int INum=0;
```

```
    while(it2.hasNext()){
      Model key2=it2.next();//eg. AB
      //System.out.println("the key is:"+key2);
      List<ModelInstance> value2=lc.get(key2);
      supTemp=1.0;// Computing the Pi values of the co-locations again
      //System.out.println("value.size():"+value2.size());
      //System.out.println("******************");
      for(int i=0;i<k;i++){
        listDistinct=new ArrayList<Integer>();
        //System.out.println("******************"+i);
        for(int j=0;j<value2.size();j++){
          sz=value2.get(j).get(i);
          //System.out.println(sz);
          if (listDistinct.contains(sz)==false){
            listDistinct.add(sz);
          }
        }//end for
        everyCount=listDistinct.size();
        //System.out.println("distinct count: "+everyCount);
        eventType=key2.get(i);
        everyOCount=eventmap.get(eventType).size();
        //System.out.println("Object count: "+everyOCount);
        supNow=everyCount/everyOCount;
        //System.out.println("supNow: "+supNow);
        if (supTemp>supNow){
          supTemp=supNow;
        }

      }//endfor
      //System.out.println("supTemp: "+supTemp);
      if (supTemp>=minSup){
        INum++;
        lk.put(key2, supTemp);
        //stmt.executeUpdate("insert into col_link values('ASP Chinese Network
','http://www.aspcn.com/')");
        String s3="insert into p"+k;
        String s4="INum,";
        String s5=INum+",";
        for(int m=1;m<=key2.size();m++){
          s4=s4+"type"+m+",";
          s5=s5+"\'"+key2.get(m-1).toString().trim()+"\'"+",";
        }
        s3=s3+" ("+s4+"pi ) values("+s5+supTemp+")";

        //System.out.println("s3="+s3);
        stmt.executeUpdate(s3);

        for(int i=0;i<value2.size();i++){
          s3="insert into t"+k;
          s4="INum,";
          s5=INum+",";
          for(int j=0;j<k;j++){
            s4=s4+"id"+(j+1);
            s5=s5+value2.get(i).get(j);
            if (j<k-1){
              s4=s4+",";
              s5=s5+",";
            }
          }
          s3=s3+" ("+s4+" ) values("+s5+")";
```

```
        //System.out.println("s3="+s3);
        stmt.executeUpdate(s3);
      }

    }

  }//end while

  return lk;
  //

}

  public void generateModel(
    LinkedHashMap<Model,List<ModelInstance>> lc,
    LinkedHashMap<Model,Double> lk,
    LinkedHashMap<Model,List<ModelInstance>> distancePair,
    int index,
    LinkedHashMap<Model,List<ModelInstance>> lc2
    ){
  // lc lk distancePair k ->lc2
  Model source,target;
  List<Model> keys=new ArrayList<Model>();/
  //System.out.println("generateModel");
  //System.out.println(lk.keySet());
  keys.addAll(lk.keySet());
  for(int i=0;i<keys.size()-1;i++){
    source=keys.get(i);
    //System.out.println("source:"+source);
      //for(int k=0;k<source.size();k++)
            //System.out.print(source.list.get(k));
      //System.out.println("");
    for(int j=i+1;j<keys.size();j++){
      target=keys.get(j);
      //System.out.println("target:"+target);

      if(source.partialEqual(target, index) ){
        Model searchModel=new Model();
        searchModel.add(source.get(source.size()-1));
        searchModel.add(target.get(source.size()-1));
        //System.out.println("searchModel:"+searchModel);
        Model newModel=new Model();
        newModel.addPartialValue(source, index);
        newModel.addPartialValue(searchModel, searchModel.size());

        int k=index+2;/

        int m=0;
        boolean xhbl=true;
              while(xhbl==true && m<k-2){
          Model zh=new Model();
          for(int n=0;n<k;n++){
            if (n!=m){

              zh.add(newModel.get(n));
            }
          }
          m++;
```

```
          if(lk.containsKey(zh)==false){

            xhbl=false;//
          }
        }//end while
        if (xhbl==true){
          lc2.put(newModel, new ArrayList<ModelInstance>());
          //Generating table-instances
          generateInstance(lc.get(source),
              lc.get(target), index,searchModel,newModel);
        }
      }
    }
  }

}
public void generateInstance(List<ModelInstance> sourceList,
    List<ModelInstance> targetList,int index,
    Model searchModel,Model newModel){
  //distancePair,index,
  //searchModel  BC,newModel  ABC

  ModelInstance sourceI,targetI;
  List<ModelInstance> distanceInstance;/
  for(int i=0;i<sourceList.size();i++){
    sourceI=sourceList.get(i);//eg 11
    for(int j=0;j<targetList.size();j++){
      targetI=targetList.get(j);//eg 12
      if(sourceI.partialEqual(targetI, index)){
        ModelInstance searchInstance,newInstance;
        searchInstance=new ModelInstance();
        newInstance=new ModelInstance();
        searchInstance.add(sourceI.get(sourceI.size()-1));
        searchInstance.add(targetI.get(targetI.size()-1));

        //System.out.println("searchI:"+searchInstance);

        distanceInstance=distancePair.get(searchModel);
        if (distanceInstance.contains(searchInstance)){
          newInstance.addPartialValue(sourceI, index);
          newInstance.addPartialValue(searchInstance, searchInstance.size());
          lc2.get(newModel).add(newInstance);

        }//endif
      }//endif

    }//endfor
  }// end for
}
public java.util.Date accountTime(java.util.Date nowTime1,String message){
  java.util.Date nowTime2=
    new java.util.Date(System.currentTimeMillis());
      long hms2 = nowTime2.getTime();
      long hms1 = nowTime1.getTime();
      long jg_hms = hms2 - hms1;
      System.out.print(message);
      //System.out.println("The last running time: " + nowTime1);
      //System.out.println("The running time of this work: " + nowTime2);
      //System.out.println("Conversing it to milliseconds: " + hms1);
      //System.out.println("Conversing it to milliseconds: " + hms2);
```

```
        System.out.println(" : " + jg_hms+"milliseconds");
        return nowTime2;
  }

  public void genPT(int k,Statement stmt) throws SQLException{

    //stmt.executeUpdate("CREATE TABLE col_link (sitename varchar (20) NULL ,siteurl varchar
(50) NULL) ");
    String createP="create table t"+k+"(INum decimal(10),";
    for(int i=1;i<=k;i++){
      createP=createP+"id"+i+" decimal(10)";
      if (i<k){
        createP=createP+",";
      }
    }
    createP=createP+")";
    //System.out.println(createP);
      ResultSet rt=stmt.executeQuery("select count(*) as n  from sysobjects where name
='t"+k+"'");
    if (rt.next()) {
      int num=Integer.valueOf(rt.getString("n"));
      if (num>0){
        stmt.executeUpdate("drop table t"+k);
      }
    }

    stmt.executeUpdate(createP);

  }
  public void genTable(Statement stmt) throws SQLException{
    String createP="";
    for(int i=1;i<=k;i++){
      createP="create table t"+i+"(INum decimal(2))";
      //System.out.println(createP);
      stmt.executeUpdate(createP);
    }

  }
  public void mainProg(double m,double d,String filename) throws IOException,SQLException{
    this.minSup=m;
    this.distance=d;
    k=2;

        java.util.Date startTime=
      new java.util.Date(System.currentTimeMillis());

    String url = "jdbc:odbc:sqlFullJoin";
    Connection con;
    Statement stmt;
    try {
      Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");/
    } catch(java.lang.ClassNotFoundException e) {
      System.err.print("ClassNotFoundException: ");/
      System.err.println(e.getMessage());//
    }
    try {
      con = DriverManager.getConnection(url, "sa", "739555");//
      stmt = con.createStatement();    //
      read(stmt,filename);
      //printEvent();
```

```
    java.util.Date t2=accountTime(startTime,"Input the data");
       java.util.Date t1=t2;

    distancePair=caculateDistance(distance);
    t2=accountTime(t1,"Computing the distances");
    t1=t2;
    genPT(2,stmt);
    lk=calculateSup(distancePair,k,stmt);
    //printSup(lk,2);
    t2=accountTime(t1,"The size-2 co-locations");
    t1=t2;
    lc=distancePair;

    for(int i=2;i<eventmap.size();i++){
      if (lk.isEmpty()){
        System.out.println("The end of the program!");
        break;
      }

      lc2=new LinkedHashMap<Model,List<ModelInstance>>();
      generateModel(lc,lk,distancePair,i-1,lc2);

      genPT((i+1),stmt);
      lk=calculateSup(lc2,i+1,stmt);/

      lc=lc2;
      t2=accountTime(t1,"The size"+(i+1)+"prevalence co-locations");
      t1=t2;
    }

    stmt.close();
    con.close();

  } catch(SQLException ex) {
  System.err.println("SQLException: " + ex.getMessage());
     }

  java.util.Date endTime=accountTime(startTime,"the whole time ");
    }

 /*
 public static void main(String args[]) throws IOException,SQLException{
   FullJoin event=new FullJoin();

   event.mainProg(0.2,3000,"event300");

 }

 */
 public static void main(String[] args)  throws IOException,SQLException {
   double r,minprev;
   String filename;
   FullJoin colocation=new FullJoin();

      if(args.length != 3){
         System.out.println("Please input parameters：The first one is the distance D, the second
is the min-prev, and the third is the name of the file ");
         return;
      }
      try{
```

```
            r = Double.valueOf(args[0]);
            minprev = Double.valueOf(args[1]);
            filename=args[2].trim();
        }catch(Exception e){
            System.out.println("The parameters inputted by you are wrong, please input them again!");
            return;
        }
        System.out.println("full join" );
        System.out.print("R：" + r);
        System.out.print("    min Prev:" + minprev);
        System.out.println("    file name:" + filename);

        colocation.mainProg(minprev,r,filename);

   }// end main

}


**************************************************************************
LessJoin.java  //Join-less algorithm
package SFullSjk;

import java.io.IOException;
import java.util.*;
import java.sql.*;

public class LessJoin {
   private LinkedHashMap<String,List<Instance>> eventmap=new

LinkedHashMap<String,List<Instance>>() ;
   private LinkedHashMap<Model,List<ModelInstance>> distancePair;
   private LinkedHashMap<String,Sneighber> starN;
   private LinkedHashMap<Model,List<ModelInstance>> lcs1;
   private LinkedHashMap<Model,List<ModelInstance>> lc;
   private LinkedHashMap<Model,Double> lks1;
   private LinkedHashMap<Model,Double> lk;
   private int iNum=0;

   double distance;
   private double minSup;

   int k;

   public void read(Statement stmt,String filename) throws

IOException,SQLException {
      eventmap=new LinkedHashMap<String,List<Instance>>();
      String eventType;
      int instanceNumber=0;
      double x=0;
      double y=0;
      List<Instance> listInstance;
      listInstance=new ArrayList<Instance>();
      String sqlStr="select * from "+filename.trim()+"

order by 1,2";
      ResultSet rs = stmt.executeQuery(sqlStr);
       while (rs.next()) {
        eventType=rs.getString(1).trim();
```

```
        String bl=rs.getString(2).trim();
        instanceNumber=Integer.valueOf(bl);
        x=Double.valueOf( rs.getString(3));
        y=Double.valueOf(rs.getString(4));

        Instance instance1=new Instance

(instanceNumber,x,y);

        if (eventmap.keySet().contains(eventType)==false){
          listInstance=new

ArrayList<Instance>();
          eventmap.put(eventType,

listInstance);
        }
        eventmap.get(eventType).add(instance1);

      }//end file
    }//end FileReadIn

   //Generating the set of the star neighborhoods
   public  LinkedHashMap<String,Sneighber> genStarN(double disParameter){

         starN=new LinkedHashMap<String,Sneighber>();
         Sneighber sneighber;
         Sevent sevent;
         List<Integer> slist;

         List<String> eventList=new ArrayList<String>();
         String eventOne,eventTwo;
         Instance instance1,instance2;
         List<Instance> instanceListOne,instanceListTwo;
         eventList.addAll(eventmap.keySet());
         for(int i=0;i<eventList.size()-1;i++){
            eventOne=eventList.get(i);
            instanceListOne=eventmap.get(eventOne);
            sneighber=new Sneighber();
            starN.put(eventOne, sneighber);
            for(int m=0;m<instanceListOne.size();m++){
              instance1=instanceListOne.get(m);
              sevent=new Sevent();
              sneighber.put(instance1.getNumber(), sevent);
              for(int j=i+1;j<eventList.size();j++){
                 eventTwo=eventList.get(j);
                 instanceListTwo=eventmap.get(eventTwo);
                 slist=new ArrayList<Integer>();
                 for(int n=0;n<instanceListTwo.size();n++){
                   instance2=instanceListTwo.get(n);
              double jl=distance

(instance1,instance2);
                    if(jl<=disParameter){
                  slist.add

(instance2.getNumber());

                    }//endif
                  }//endfor instanceListTwo
```

```
                if (slist.size()>0)
                {
                  sevent.put(eventTwo, slist);
                }
              }//end for eventList
           }//end for eventList
         }//end for
         return starN;
      }


   public double distance(Instance instance1, Instance instance2 ){
     double result;
     result=Math.sqrt(
     (instance1.getX()-instance2.getX())*
     (instance1.getX()-instance2.getX())+
     (instance1.getY()-instance2.getY())*
     (instance1.getY()-instance2.getY()) );
     return result;
   }


  public void printEvent(){
    Set s1=eventmap.keySet();
    Iterator<String> itEvent=s1.iterator();
    while(itEvent.hasNext()){
      String event=itEvent.next();
      System.out.println("the key is:"+event);
      List<Instance> listinstance1=eventmap.get(event);
      System.out.println("listinstance.size

():"+listinstance1.size());
      Iterator<Instance>

itInstance=listinstance1.iterator();
      while(itInstance.hasNext()){
        System.out.println(itInstance.next

().toString());
      }

    }

  }

  public void printPK(LinkedHashMap<Model,List<ModelInstance>> pk,int i,String message){
    System.out.println(message+i+"The beginning of printing");
    Set s1=pk.keySet();
    Iterator<Model> it=s1.iterator();
    while(it.hasNext()){
      Model model=it.next();
      List<ModelInstance> listinstance1=pk.get(model);
      System.out.print(model);
      System.out.println(" length="+listinstance1.size());
      Iterator<ModelInstance>

itInstance=listinstance1.iterator();
      while(itInstance.hasNext()){
        System.out.println(itInstance.next
```

```
().toString());
      }

    }
    System.out.println(message+i+"The end of printing");

  }
  //Printing the Pi values
  public void printSup(LinkedHashMap<Model,Double> lk,int i){
    System.out.println("The Pi value of"+i+"co-location:");
    System.out.println(lk.keySet().toString());
    System.out.println(lk.values().toString());
  }
    //p1
  public LinkedHashMap<Model,Double> p1(){
    lk=new LinkedHashMap<Model,Double>();

    Model model1;
    List<String> eventList=new ArrayList<String>();
        eventList.addAll(eventmap.keySet());
        String event1;
        for(int i=0;i<eventList.size();i++){
            event1=eventList.get(i);
          model1=new Model();
          model1.add(event1);
      lk.put(model1, 1.0);
    }

        return lk;

  }
  public  void genP( int k,Statement stmt) throws SQLException

{


    //stmt.executeUpdate("CREATE TABLE col_link (sitename varchar (20) NULL ,siteurl varchar
(50) NULL) ");
    String createP="create table pl"+k+"(INum decimal(10),";
    for(int i=1;i<=k;i++){
      createP=createP+"type"+i+" char(10),";
    }
      ResultSet   rt=stmt.executeQuery("select count(*) as n " +
          "from sysobjects where name  ='pl"+k+"'");
    if (rt.next()) {
      int num=Integer.valueOf(rt.getString("n"));
      if (num>0){
        stmt.executeUpdate("drop table pl"+k);
      }
    }
    createP=createP+"pi decimal(10,2))";
    stmt.executeUpdate(createP);
  }
  public  void supExact( int k,Statement stmt,double supTemp,Model key,
      List<ModelInstance> value)  throws SQLException

{
    if (supTemp>=minSup){
      iNum++;
      lc.get(key).addAll(value);
```

```
      lk.put(key, supTemp);
    String s3="insert into pl"+k;
      String s4="INum,";
      String s5=iNum+",";
      for(int m=1;m<=key.size();m++){
        s4=s4+"type"+m+",";
        s5=s5+"\'"+key.get(m-1).toString().trim()

+"\'"+",";
      }
      s3=s3+" ("+s4+"pi ) values("+s5+supTemp+")";
      stmt.executeUpdate(s3);

      for(int i=0;i<value.size();i++){
        s3="insert into tl"+k;
        s4="INum,";
        s5=iNum+",";
        for(int j=0;j<k;j++){
          s4=s4+"id"+(j+1);
          s5=s5+value.get(i).get(j);
          if (j<k-1){
            s4=s4+",";
            s5=s5+",";
          }
        }
        s3=s3+" ("+s4+" ) values("+s5+")";
        stmt.executeUpdate(s3);
      }
    }

  }

  public  double sup( int k,Model key,List<ModelInstance> value){
    List listDistinct;
    Integer sz;
    double supTemp=1.0;
    double supNow=0.0;
    double everyCount=0.0;
    double everyOCount=0.0;
    String eventType;
  if(value.size()==0){
      supTemp=0.0;
      return supTemp;
    }else{

      for(int i=0;i<k;i++){

        listDistinct=new ArrayList<Integer>();

        for(int j=0;j<value.size();j++){

          sz=value.get(j).get(i);

          if (listDistinct.contains(sz)

==false){
            listDistinct.add(sz);

          }
```

```
      }//end for


      everyCount=listDistinct.size();


      eventType=key.get(i);
      everyOCount=eventmap.get(eventType).size();
      supNow=everyCount/everyOCount;

      if (supTemp>supNow){

        supTemp=supNow;
      }

    }//endfor
    return supTemp;
  }//end if

 }
 public  boolean supCoarse( Model key,double supTemp){
  if (supTemp<minSup){

    lc.remove(key);
    //System.out.println("coarse delete!"+key);
    return true;
  }else{
    return false;
  }

 }
 public List<ModelInstance> filterClique(int k,Model

key,List<ModelInstance> value){
                    //  3  abc,111 242 331  lcs1
  Model checkKey=new Model();
  ModelInstance checkValue;
  List<ModelInstance> valueNew=new ArrayList<ModelInstance>();
  List<ModelInstance> cliqueValue=new

ArrayList<ModelInstance>();
  for(int i=1;i<k;i++){
    checkKey.add(key.get(i));
  }

  cliqueValue=lcs1.get(checkKey);
  for(int i=0;i<value.size();i++){
    checkValue=new ModelInstance();
    for(int j=1;j<value.get(i).size();j++){
      checkValue.add(value.get(i).get(j));
    }
    if (cliqueValue.contains(checkValue)==true){
      valueNew.add(value.get(i));
    }
  }
  return valueNew;
 }

 public void genModel(int index ){
```

```
   Model source,target;
   List<Model> keys=new ArrayList<Model>();
   keys.addAll(lks1.keySet());
   for(int i=0;i<keys.size();i++){
     source=keys.get(i);
     for(int j=i+1;j<keys.size();j++){
       target=keys.get(j);
       if(source.partialEqual(target, index) ){
         Model searchModel=new Model();
         searchModel.add(source.get

(source.size()-1));
         searchModel.add(target.get

(source.size()-1));
         //System.out.println

("searchModel:"+searchModel);
         Model newModel=new Model();
         newModel.addPartialValue(source,

index);
         newModel.addPartialValue

(searchModel, searchModel.size());


         int k=index+2;

         int m=0;
         boolean xhbl=true;
         while(xhbl==true && m<k-2){
           Model zh=new Model();
           for(int n=0;n<k;n++){
             if (n!=m){

               zh.add

(newModel.get(n));
             }
           }
           m++;

           if(lks1.containsKey(zh)

==false){

//System.out.println(newModel);
//System.out.println(zh);
             xhbl=false;/
           }
         }//end while
         if (xhbl==true){
           lc.put(newModel, new

ArrayList<ModelInstance>());
         }
       }
     }
   }
```

```
 }

 public void genInstance(int k,Statement stmt) throws SQLException{
  // lc(k)+starN->Instance of lc(k)
  List<Model>  keys;
  List<Integer> keys2;
  Sevent sevent;
  List<ModelInstance> list1,list3,list;
  List<Integer> list2;
  ModelInstance id1;
  keys=new ArrayList<Model>();
  keys.addAll(lc.keySet());
  int centerId;

  for(int i=0;i<keys.size();i++){
   Model model=keys.get(i);
   //iNum++;

   String center,neighber;
   center=model.get(0);
   Sneighber sneighber=starN.get(center);

   keys2=new ArrayList<Integer>();
   keys2.addAll(sneighber.map.keySet());
   list=new ArrayList<ModelInstance>();
   for(int n=0;n<keys2.size();n++){
    centerId=keys2.get(n);//1
    id1=new ModelInstance();
    id1.add(centerId);
    list1=new ArrayList<ModelInstance>();
    list3=new ArrayList<ModelInstance>();
    list1.add(id1);
    sevent=sneighber.get(centerId);
    boolean kg=true;
    for(int j=1;j<model.size();j++){
     neighber=model.get(j);

     if (sevent.containsKey(neighber)){
      //if (center.equals("AA") &&

neighber.equals("CC"))
       //{System.out.println

(neighber);}
       list2=sevent.get(neighber);
       //list1*list2->list3
       list3=genInstance2

(list1,list2);
       list1=list3;
      }
      else{
       kg=false;
       break;
      }
    }//end keys2
    //model list3
    if(kg==true){
     list.addAll(list3);
```

```
      }
    }//end model
    if(k>2){

      double sup1=sup(k,model,list);
      if (supCoarse(model,sup1)==false){

        list=filterClique(k,model,list);

        double sup3=sup(k,model,list);
        supExact( k,stmt,sup3,model,list);
      }
    }else{

      double sup2=sup(k,model,list);
      supExact( k,stmt,sup2,model,list);
    }
  }// end for
}

  public List<ModelInstance> genInstance2(List<ModelInstance>, list1,List<Integer> list2){
    List<ModelInstance> list=new ArrayList<ModelInstance>();
    ModelInstance modelI;
    int id;
    for(int i=0;i<list1.size();i++){
      for(int j=0;j<list2.size();j++){
        modelI=new ModelInstance();
        modelI.addAll(list1.get(i));
        id=list2.get(j);
        modelI.add(id);
        list.add(modelI);
      }// end list2
    }//end list1
    return list;
  }

  public java.util.Date accountTime(java.util.Date nowTime1,String, message){
    java.util.Date nowTime2=
      new java.util.Date(System.currentTimeMillis());
        long hms2 = nowTime2.getTime();
        long hms1 = nowTime1.getTime();
        long jg_hms = hms2 - hms1;
        System.out.print(message);

        System.out.println(" : " + jg_hms+"milliseconds");
        return nowTime2;
  }

  public void genT(int k,Statement stmt) throws SQLException{

    //stmt.executeUpdate("CREATE TABLE col_link (sitename
varchar (20) NULL ,siteurl varchar (50) NULL) ");
    String createP="create table tl"+k+"(INum decimal(10),";
    for(int i=1;i<=k;i++){
      createP=createP+"id"+i+" decimal(10)";
      if (i<k){
        createP=createP+",";
      }
    }
    createP=createP+")";
```

```
      ResultSet rt=stmt.executeQuery("select count(*) as n

from sysobjects where name ='tl"+k+"'");
   if (rt.next()) {
     int num=Integer.valueOf(rt.getString("n"));
     if (num>0){
       stmt.executeUpdate("drop table tl"+k);
     }
   }

   stmt.executeUpdate(createP);

 }
 public void mainProg(double m,double d,String filename) throws

IOException,SQLException{
   this.minSup=m;
   this.distance=d;
      java.util.Date startTime=
     new java.util.Date(System.currentTimeMillis());

   String url = "jdbc:odbc:sqlFullJoin";

Sqldsn is the name of the dsn
   Connection con;
   Statement stmt;
   try {
     Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
   } catch(java.lang.ClassNotFoundException e) {
     System.err.print("ClassNotFoundException: ");
     System.err.println(e.getMessage());/
   }
   try {
     con = DriverManager.getConnection(url, "sa",

"739555");
     stmt = con.createStatement();

     read(stmt,filename);//printEvent();//
     java.util.Date t2=accountTime(startTime,"Input the data");
        java.util.Date t1=t2;
     starN=genStarN(distance);//
     t2=accountTime(t1,"generating the set of the star neighborhoods");
     t1=t2;
       lks1=p1();

     for(int k=2;k<=eventmap.size();k++){
       if (lks1.isEmpty()){
         System.out.println("The end of the program!");
         break;
       }

       lc=new

LinkedHashMap<Model,List<ModelInstance>>();
       lk=new LinkedHashMap<Model,Double>();
       genModel(k-2);
       genP( k,stmt); genT(k,stmt);
       genInstance(k,stmt);
       //printPK(lc,k,"The set of the candidate co-locations ");
```

```
        //printSup(lk,k);
        lks1=lk;
        lcs1=lc;
        t2=accountTime(t1,"The size"+k+"co-locations");
        t1=t2;
      }
      stmt.close();
      con.close();

    } catch(SQLException ex) {
    System.err.println("SQLException: " + ex.getMessage());

    }
    java.util.Date endTime=accountTime(startTime,"the whole time");
  }


  public static void main(String args[]) throws

  public static void main(String[] args)  throws

IOException,SQLException {
    double r,minprev;
    String filename;
    LessJoin colocation=new LessJoin();
        if(args.length != 3){
            System.out.println("Please input the parameters: The distance D, the min-prev and the
name of the file. ");
            return;
        }
        try{
            r = Double.valueOf(args[0]);
            minprev = Double.valueOf(args[1]);
            filename=args[2].trim();
        }catch(Exception e){
            System.out.println("Your input is wrong, please input again!");
            return;
        }
        System.out.println("less join" );
        System.out.print("R：" + r);
        System.out.print("    min Prev:" + minprev);
        System.out.println("    file name:" + filename);

        colocation.mainProg(minprev,r,filename);

  }// end main

}
```

Appendix 2

The Papers Published during this Doctor of Philosophy Degree:

AOG-ags Algorithms and Applications*

Lizhen Wang[1,2], Junli Lu[1], Joan Lu[2], Jim Yip[2]

[1] Department of Computer Science and Engineering, School of Information, Yunnan University, Kunming, 650091, P. R. China
Lzhwang@ynu.edu.cn
[2] Department of Informatics, School of Computing and Engineering, University of Huddersfield, Huddersfield, UK, HD1 3DH

Abstract. The attribute-oriented generalization (AOG for short) method is one of the most important data mining methods. In this paper, a reasonable approach of AOG (AOG-ags, attribute-oriented generalization based on attributes' generalization sequence), which expands the traditional AOG method efficiently, is proposed. By introducing equivalence partition trees, an optimization algorithm of the AOG-ags is devised. Defining interestingness of attributes' generalization sequences, the selection problem of attributes' generalization sequences is solved. Extensive experimental results show that the AOG-ags are useful and efficient. Particularly, by using the AOG-ags algorithm in a plant distributing dataset, some distributing rules for the species of plants in an area are found interesting.

Keywords: Attribute-oriented generalization (AOG); Concept hierarchy trees; Attributes' generalization sequences (AGS); Equivalence partition trees; Interestingness of AGS.

1 Introduction

The general idea of Attribute-oriented generalization (AOG) is to abstract each attribute of each record in a relation from a relatively low conceptual level to higher conceptual levels by using domain knowledge(Concept Hierarchy Trees), in order to discover rules among attributes from multilevel or higher lever. Two AOG algorithms are well-known: AOI [1] and LCHR [2]. Both are not incremental and also don't allow fast re-generalization. An AOG method possessing fast re-generalization was proposed in literature [3]. However, it is not perfect in efficiency and occupies too much memory space.

C. L. Carter and H. J. Hamilton proposed two new AOG algorithms (the GDBR and the FIGR) in literature [4]. The GDBR is an online algorithm, while the FIGR has characteristics of incremental and fast re-generalization. One important thing is that the runtime of the GDBR and the FIGR is less than the AOI and the LCHR. But there is a supposition in the FIGR. That is that the size of attributes and the number of the possible values in an attribute are relatively small. In addition, the four algorithms

* Supported by the National Natural Science Foundation of China under Grant No.60463004.

control generalization levels by using attribute thresholds. That is not so practical in some applications, because it is impossible to try every possible combination of thresholds for every attribute, and the size of attributes and the number of the possible values of an attribute are not small in some practical applications.

Considering the generalization threshold, besides the attribute thresholds, Chen [5] and Han [6] discussed the degree of tuple (record) generalization (i.e. tuple thresholds), and the problem of combining the attribute thresholds and the tuple thresholds. Under the tuple thresholds, it is convenient to control the process of AOG. But the low-efficiency of AOG algorithms becomes the main problem in its applications, for we have to consider all combinations of attributes which satisfy the tuple threshold in an attribute-oriented generalization process, and explain the results of generalization (or rank them for user).

In this paper, by introducing the concept of the attributes' generalization sequence, the attribute thresholds and the tuple thresholds are unified, and a reasonable approach of AOG—AOG-ags (Attribute-Oriented Generalization based on Attributes' Generalization Sequences), which expands the traditional AOG efficiently, is proposed. Some technologies, for example, partitions, equivalence partition trees, prune optimization strategies and interestingness, are used to improve the efficiency of the algorithm. It is shown that the AOG-ags algorithm has special advantages.

The rest of the paper is organized as following. Section 2 formally defines the concept of the degree of tuple generalization, and introduces the method of AOG based on attributes' generalization sequence (AOG-ags). In Section 3, by introducing the equivalence partition trees, an optimization algorithm of AOG-ags is devised. Interestingness of attributes' generalization sequences is discussed in Sect. 4. Section 5 discusses correctness, completeness and complexity of our algorithms. Performance and application results of algorithms are evaluated in Sect. 6. The last Section is conclusions.

2 AOG-ags

In traditional AOG algorithms, the generalization process is controlled by setting a threshold for each attribute. But in some applications, user does not want to consider each attribute for generalization threshold, so the degree of tuple generalization is introduced.

Definition 1 Given a relation $r(r_1,\ldots,r_n)$ and the generalization relation $r'(r_1',\cdots,r_n')$, then the rate of reserved tuples is defined as $\overline{Z}=n'/n$, so the degree of tuple generalization is defined as $Z=1-\overline{Z}=1-(n'/n)$.

Z is a measure for the degree of tuple generalization. The higher is the value of Z, the greater the degree of generalization. The value Z meets $0\le Z\le(n-1)/n$.

Z cannot confirm certain generalization result. That is to say, given a tuple threshold Z, we will get some generalization relations that satisfy this threshold Z. But analyzing the process of AOG, we find that generalization for each attribute is independ-

ent, that is, an attribute is generalized earlier or latter will not affect the generalization result. Further to say, generalization result is the same no matter that it is obtained by generalizing gradually or directly up to the k-th level, so attributes' generalization sequences (AGS for short)is introduced in this research. One AGS confirms certain generalization relation.

The Depth of certain node V in the tree is defined as the path length from root to V, the Height of V is the length of the longest path in the tree whose root is V. The Height of the tree is the Height of its root. The Level of V is its Depth more 1.

Definition 2 Given a relation pattern $R(A_1,\cdots,A_m)$, attributes' concept hierarchy trees $h_1,\cdots,h_m$, the Heights of trees $l_1,\cdots,l_m$, sequence $A_1^{g_1}\cdots A_i^{g_i}\cdots A_m^{g_m}$ is called an AGS of AOG, where ($1\le g_i\le l_i+1$).

Property 1 The number of all AGS in a relation pattern is $\prod_{i=1}^{m}(l_i+1)$。

Proof. ∵ One sequence $g_1\cdots g_i\cdots g_m$ can only confirm one AGS $A_1^{g_1}\cdots A_i^{g_i}\cdots A_m^{g_m}$.

Meanwhile, ∵ $1\le g_i\le l_i+1$ ∴ $|g_i|=l_i+1\quad(1\le i\le m)$

∴ The number of attributes' generalization sequences is:

$$(l_1+1)\times\cdots\times(l_i+1)\times\cdots\times(l_m+1)=\prod_{i=1}^{m}(l_i+1)\qquad\square$$

Definition 3 Given the tuple threshold Z. If the generalization relation $r'(r_1',\ldots r_{n'}')$ which are confirmed by the AGS $A_1^{g_1}\cdots A_i^{g_i}\cdots A_m^{g_m}$ ($1\le g_i\le l_i+1$) satisfies $1-(n'/n)\ge Z$, and if increasing any $g_i(1\le i\le m)$, it will not satisfy $1-(n'/n)\ge Z$, then $A_1^{g_1}\cdots A_m^{g_m}$ is called an AGS which satisfies the Z, and $r'(r_1',\ldots r_{n'}')$ is called a generalization result under $A_1^{g_1}\cdots A_m^{g_m}$.

From Definition 3, we can conclude that the AOG method of using attribute thresholds is a special case of using the tuple thresholds. That is to say, the attribute thresholds and the tuple thresholds are unified under the concept of AGS. The AOG based on AGS (for short, call it AOG-ags) is an efficient extension to the traditional approach.

An ordinary AOG-ags algorithm is devised as follows.

```
Algorithm 1: The ordinary AOG-ags algorithm
Input: The un-generalized dataset (relation) r, which has m
attributes{ A1,A2….Am },Attributes' concept hierarchy
trees {h1,⋯,hm} and the height of trees {l1,⋯,lm} ,The tuple
threshold Z
Output: Generalization rules which meet the Z
Algorithm description:
1) Gen_seq(relation,1,m,L1,S,Gs);
```

```
2) Selecting a sequence from the set Gs of AGS and returning
a generalization relation;
3) Producing generalization rules from the generalization re-
lation.

Procedure Gen_seq(r,i,m,Li,S,Gs);
(1)   For k=Li+1 downto 1 Do
(2)     Begin If k< Li+1 then
(3)              Gen_r ←generalize (r,i,k)
(4)           Else  Gen_r ←r   Endif;
(5)           If i<m then
(6)              Gen_seq (Gen_r,i+1,m,Li+1,S∪Aki,Gs)
(7)           Else
(8)             If |Gen-r|≤n(1-Z) then
(9)               Gs ←Gs∪ {S∪Aki}
                endif
              Endif
        End
```

When the number of attributes (m) is larger, in order to obtain all attributes' generalization sequences which meet the Z in algorithm, we must search $\prod_{i=1}^{m}(l_i+1)$ times, and it will waste much time. So, how to efficiently compute all AGS which meet the Z is the chief problem in this algorithm. Further more, how to quickly calculate generalization relations that are related to AGS is another solving problem. To solve these problems efficiently, an optimization AOG-ags algorithm is presented by introducing equivalence partition trees according to the property of AGS.

3 An Optimization AOG-ags Algorithm

3.1 AOG-ags and partition

Let $r(r_1,\ldots,r_n)$ is a relation in relation pattern $R(A_1,\cdots,A_m)$, $X\subseteq R$, $\forall r_i,r_j\in r$, r_i and r_j are equal with respect to X if and only if $r_i[A_k]=r_j[A_k]$ for $\forall A_k\in X$, which is denoted as $r_i=_X r_j$. X partitions the rows of r into equivalence classes. We can denote the equivalence class of $r_i\in r$ with respect to $X\subseteq R$ by $[r_i]_{=X}$. The quotient set $\pi_x=\{[r_i]_{=x}\mid r_i\in r\}$ of equivalence classes is an equivalence partition of r under X.

Given two partitions $\pi_x=\{\tau_1,\cdots,\tau_k\}$, $\pi_y=\{\tau'_1,\cdots,\tau'_k\}$, $\pi_X\otimes\pi_Y=\{\tau_i\cap\tau'_j\mid\tau_i\in\pi_x,\tau'_j\in\pi_y\}$ is called intersection partition of π_X and π_Y, $\pi_X\otimes\pi_Y$ is a partition of r. We know that $\pi_{X\cup Y}=\pi_X\otimes\pi_Y$ holds.

In fact, the equivalence class $\pi_R=\pi_{\{A_1,\cdots,A_m\}}$ is the relation r. According to the property of $\pi_{X\cup Y}=\pi_X\otimes\pi_Y$, there is a one-one correspondence from the records of r to equivalence classes of $\pi_{\{A_1\}}\otimes\cdots\otimes\pi_{\{A_m\}}$. If $\pi_{A_i,j}(1\le i\le m,1\le j\le l_i+1)$ denotes equivalence partition which attribute A_i generalizes up to the j-th level along with the concept

hierarchy tree, then the generalization relation and AGS $A_1^{g_1}\cdots A_m^{g_m}$ is corresponding one by one. This leads to a new partition-based approach of AOG-ags:

(1) Compute all $\pi_{A_i,g_i}(1\le i\le m,1\le g_i\le l_i+1)$.
(2) Obtain all AGS which meet the Z.
(3) Select a sequence $A_1^{g_1}\cdots A_m^{g_m}$, and then calculate generalization relation $r'=\pi_{A_1g_1}\otimes\cdots\otimes\pi_{A_mg_m}$.
(4) Produce generalization rules from the generalization relation.

3.2 Pruning Strategies

Definition 4 The Grid that is constituted by $\prod_{i=1}^{m}(l_i+1)$ possible AGS and satisfies the following properties is called the search space.

(1) There are AGS that satisfy $g_1+\cdots+g_m=k+1$ in the k-th level.

(2) Each sequence is connected to any sequence $A_1^{g_1}\cdots A_i^{g_i-1}\cdots A_m^{g_m}$ of the (k-1)-th level.

The search space will increase with the increasing of m and l_i. By introducing the concept "refinement", some pruning is executed for reducing search space.

Definition 5 Given a relation and its two partitions $\pi_x=\{\tau_1,\cdots,\tau_k\}$, $\pi_y=\{\tau_1',\cdots,\tau_{k'}'\}$, if $\forall\tau_i\in\pi_X,\exists\tau_j'\in\pi_Y,\ \tau_i\subseteq\tau_j'$ holds, then π_X is called as a refinement of π_Y.

Obviously, $\pi_X\otimes\pi_Y$ is the refinement of π_X and π_Y, and $\pi_{A_i,j}$ refines $\pi_{A_i,k}$, $1\le i\le m,1\le k<j\le l_i+1$. If π_X refines π_Y, then $|\pi_X|\ge|\pi_Y|$ holds.

Definition 6 Given two sequences $A=A_1^{g_1}\cdots A_m^{g_m}$ and $A'=A_1^{g_1'}\cdots A_m^{g_m'}$, if $\forall A_i,\ g_i\ge g_i'(1\le i\le m)$ holds, then A is called a sub-sequence of A', denote as $A=sq(A')$, and A' is the parent-sequence of A, denote as $A'=fq(A)$.

If $A=sq(A')$, then π_A refines $\pi_{A'}$. Therefore, what the pruning strategies we can get are the followings.

(1) If there exists a π_{A_i,g_i}, and $|\pi_{A_i,g_i}|>n(1-Z)$ holds, then any sequence which includes $A_i^{g_i}$ or its sub-sequence $A_i^{g_k}$ ($g_k\ge g_i$) cannot meet the Z.

(2) If there is a sequence $A_i^{g_i}\cdots A_j^{g_j}$, and $|\pi_{A_i,g_i}\cup\cdots\cup\pi_{A_j,g_j}|>n(1-Z)$ holds, then any sequence which includes $A_i^{g_i}\cdots A_j^{g_j}$ or its sub-sequence may not meet the Z.

(3) If a sequence $A=A_1^{g_1}\cdots A_m^{g_m}$ meets the Z, then all parent-sequences of A will not meet the Z, so it can be pruned.

3.3 Calculating π_{A_i,ε_i}

By introducing the concept of equivalence partition trees, π_{A_i,ε_i} can be calculate efficiently. At first, we assign each node (i.e., concept) in a concept hierarchy tree to a concept's code. It is shown as the numbers of bracket in Fig. 1. The unary concept's codes represent the first level in the concept hierarchy tree (i.e., the root of the tree), ("3" is the code of the root, which only represents the difference from other attributes). The binary codes represent the second level, etc.

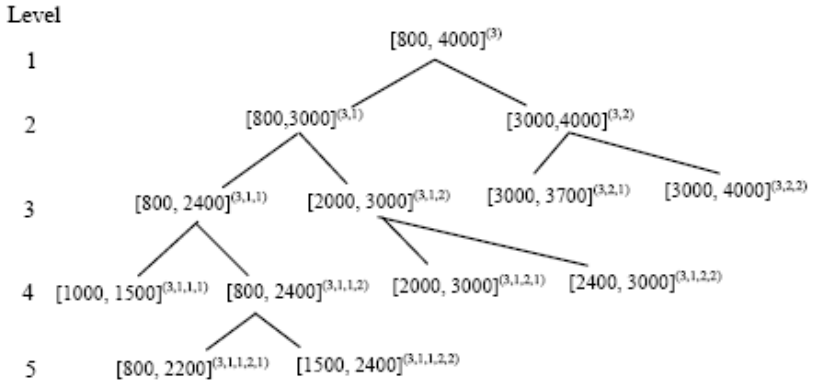


Fig. 1. A concept hierarchy tree of an attribute "elevation" and its concept's codes

Definition 7 The equivalence partition tree of the attribute A, each branch is a concrete value of A, which is concept's code with respect to the value in the concept hierarchy tree, each node in the tree is a value in concept's code.

The equivalence partition tree of the attribute "elevation", which values are from the Table 1 in Section 6.2, is showed as Fig. 2.

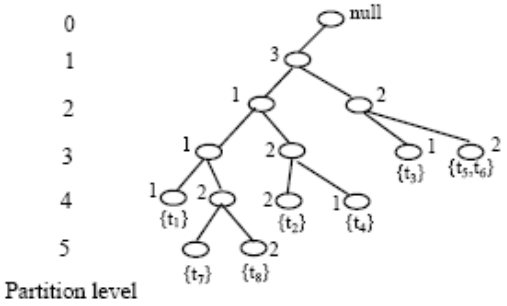


Fig. 2. The equivalence partition tree of the attribute "elevation" in Table 1

An equivalence partition tree can be constructed by the following steps: (1) create the root of the tree, labeled with "null". (2) for each value of the attribute, a branch is created. (3) note down the corresponding Row-Id under the corresponding leaf node. The set of the Row-Id noted down is called the identity of leaf-node.

The partition level of an equivalence partition tree is defined as follows. The root node "null" is the level 0, the following is level 1, …, till the leaf nodes. The identity of a node in an equivalence partition tree is the union of the identity of all leaf-nodes

of the sub-trees with this node as their root. The set of equivalence partition class with respect to attribute A at level l is the set of the identity of nodes of the l-th partition level in equivalence partition tree with respect to the attribute A.

Considering the equivalence partition tree in Figure 3, the partition result in the third level is $\{\{t_1,t_7,t_8\}, \{t_2,t_4\}, \{t_3\}, \{t_5,t_6\}\}$, which is the same as the equivalence partition result after "elevation" is generalized to the third level.

3.4 An optimization AOG-ags algorithm

Introducing partition and refinement, we can efficiently speed the AOG process up. Introducing equivalence partition trees, we can quickly obtain equivalence partition results of attributes in any level. So we have an optimization algorithm of AOG-ags as below:

Algorithm 2: An optimization algorithm of AOG-ags
Input: The un-generalized dataset r, which has m attributes $\{A_1,A_2 \ldots A_n\}$; Attribute's concept hierarchy tree $^{\{h_1,\cdots,h_m\}}$ and their heights $^{\{l_1,\cdots,l_m\}}$; The tuple threshold Z
Output: generalization rules which meet the Z
Algorithm description:
1) creation_partition_tree (r);
2) computing lower bound $L(A_1)$ which attribute A_1 is generalized;
3) Gen($\pi_{A_1,L(A_1)}$,1,m,$L(A_1)$,S,Gs); //the initial values are S="null", Gs=Φ. Obtain all AGS (Gs) which meet the Z
4) Selecting a generalization sequence $A_1^{g_1} \cdots A_m^{g_m}$ from Gs, computing generalization relation $r' = \pi_{A_1 g_1} \otimes \cdots \otimes \pi_{A_m g_m}$;
5) Producing generalization rules from the generalization relation.

```
Procedure Gen(r,i,m,L(A1),S,Gs);
(1)    For k= L(A1) downto 1 Do
(2)       Begin  If i=1 and k < L(A1)   then
(3)                 Gen_r ←π A_i,k
(4)              Else  If i=1 and k= L(A1) then
(5)                       Gen_r ← r
(6)                    Else  Gen_r ← r ⊗ π A_i,k   Endif
                 Endif;
(7)              if | Gen_r| > n(1-Z) then
(8)                exit for
                 endif
(9)              If i<m then
(10)               Gen (Gen_r,i+1,m,L(A i+1),S∪ A^k 1,Gs)
(11)             Else  If |Gen-r|≤ n(1-Z) then
(12)                     Gs ←Gs∪ {S∪ A^k 1};
(13)                     Exit for
                       Endif
```

```
                    Endif
            End
```

In the optimization algorithm of AOG-ags, we efficiently control recursive times by computing each attribute's lower limit $L(A_i)$, and consider whether AGS can be pruned or not in a recursive process according to Pruning Strategies. In fact, many recursive steps will be jumped over using pruning strategies in algorithm 2.

4 Interestingness of AGS

Motivation example: For the plant "Magnolia sieboldii" in a plant distributing dataset, suppose the following rules have been obtained.

(1) Plant "Magnolia sieboldii" ⇒ 50% grows in the conifer forest and scrub whose elevation is from 2600 to 4100 meter of Lijiang, and 50% grows in the forest, scrub and meadow whose elevation is from 2400 to 3900 meter of Weixi.

(2) Plant "Magnolia sieboldii" ⇒ 90% grows in the conifer forest and scrub whose elevation is from 2600 to 4100 meter of Lijiang, and 10% grows in the forest, scrub and meadow whose elevation is from 2400 to 3900 meter of Weixi.

The rule (2) is more meaningful than the rule (1), because the growth characteristics of plant "Magnolia sieboldii" are more obvious in the rule (2).

Definition 8 In generalization relation, the t-weight of the i^{th} generalization record t_i is defined as formula (1).

$$t_i = \frac{count\ (i)}{\sum_{j=1}^{n'} count\ (j)} \tag{1}$$

In formula (1), count (i) is the number of repeated records of the i^{th} generalization record in generalization relation, n' is the number of records in generalization relation.

Definition 9 Given $r'(r'_1,\ldots r'_{n'})$ is a generalization relation under $A_1^{g_1}\cdots A_m^{g_m}$, $(1 \le g_i \le l_i + 1)$, then interestingness $I_{g_1\cdots g_m}$ of $A_1^{g_1}\cdots A_m^{g_m}$ is defined as formula (2).

$$I_{g_1\cdots g_m} = \sqrt{\sum_{i=1}^{n'}(t_i - \frac{1}{n'})^2} \tag{2}$$

When the number of repeated records for each generalization record in a generalization relation $r'(r'_1,\ldots r'_{n'})$ gets average value, $I_{g_1\cdots g_m}$ achieves the minimum 0. The farer t-weight of generalization records in a generalization relation from average value, the larger the contribution to interestingness. The larger the value of $I_{g_1\cdots g_m}$, the more interesting the rule expressed by the attributes' generalization sequence $A_1^{g_1}\cdots A_m^{g_m}$.

Therefore, after obtaining sequences which meet the Z, computing their interestingness, and ranking the generalization sequences with the decline of interestingness, we can produce generalization relation and rules.

5 Analysis

In this section, we analyze our algorithms for completeness, correctness and computational complexity. Correctness means that the generalization rules meet the user specified threshold. Completeness implies that no AGS that satisfies the given threshold Z is missed.

– **The algorithm 1 is correct.**

Proof. The algorithm 1 uses very simple way to get generalization rules. It is obvious that this algorithm is correct if we can prove the recursive procedure Gen_seq is correct. That means the Gen_seq will return the AGS that satisfy the Z. It is guaranteed by step (8) in Gen_seq, because this step will check whether every sequence satisfies the Z or not.

– **The algorithm 2 is correct.**

Proof. The pruning strategy (1) guarantee the step 2) in algorithm 2 will return the low boundary of every attributes. If the return of step 3) is correct, then the Section 3.2 ensures the generalization relation computed by the step 4) is correct.

For step 3) (i.e., the recursive procedure Gen), the property of $\pi_{X\cup Y} = \pi_X \otimes \pi_Y$ and Section 3.2 ensure the correctness of the step (3)-(6) in Gen. The step (3)-(6) ensures the correctness of the step (11). And the step (11) guarantees every AGS satisfies the threshold Z.

– **The algorithms are complete.**

Proof. We prove if a sequence satisfies the Z, it is found by our algorithms. In the recursive procedure Gen_seq of algorithm 1, the step (1) iterates all generalization levels of an attribute and the step (6) recursively perform Gen_seq. So the combine of step (1) and step (6) ensures the Gen_seq will check all possible candidate sequences.

For algorithm 2, the pruning strategy (1) guarantee the step 2) of algorithm will return the low boundary of every attributes. In the recursive procedure Gen, The step (7) and (8) is because the pruning strategy (2) and the step (11) and (13) are just because the pruning strategy (3). The combine of step (1) and step (10) guarantee the Gen will check all possible candidate sequences.

– **Computational Complexity Analysis**

Suppose the number of records in the relation T is N, the number of attributes in T is m, and the height of attribute i-th concept hierarchy tree is l_i. In the worse case, the computational complexity of the algorithm 1 will be $O(N*\prod_{i=1}^{m}(l_i+1))$. And the algorithm 2 will be $O(\prod_{i=1}^{m}(l_i+1))$. Theoretically speaking, the computational complexity of the two algorithms seems not very distinctive. But we will jump over many recursive steps using pruning strategies in algorithm 2, so the complexity will be much lower than algorithm 1. We will show the real execution results in the experiments.

6 Performance Evaluation and Applications

The performance of our algorithms is evaluated by synthetic datasets and a real dataset (a plant distributing dataset in "The Three Parallel Rivers in Yunnan Area" zone). The experiments are performed on a Celeron computer with a 2.40 GHz CPU and 256 Mbytes memory running the Windows XP operating system.

6.1 Evaluation Using Synthetic Datasets

The experiments using synthetic data sets are aimed at answering the following questions. (1) How does the size of dataset affect the two algorithms? (2) How do the Algorithm 1 and the Algorithm 2 behave with the Z is changed?

We ran a series of experiments with increasing number of spatial data points. The results are showed in Fig. 3. (a). We can see that the algorithm 2 is almost linear and much faster than the algorithm 1.

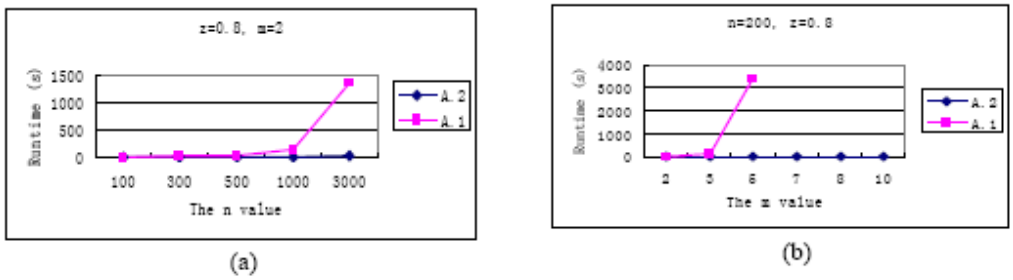


Fig. 3. Performance of algorithms using synthetic datasets

Fixed on the number of records, the number of attributes is an important parameter. The detailed comparative results are showed as Fig. 3. (b). We can see that the performance of Algorithm 1 is very bad when m=5. The results indicate that the pruning strategies and equivalence partition trees used in Algorithm 2 are very effective.

Now we look at the characters of fast re-generalization of the two algorithms. The results are shown in Fig. 4(a) (b). We used 6 different settings of the thresholds Z. We can see the Algorithm 2 possesses the character of fast re-generalization.

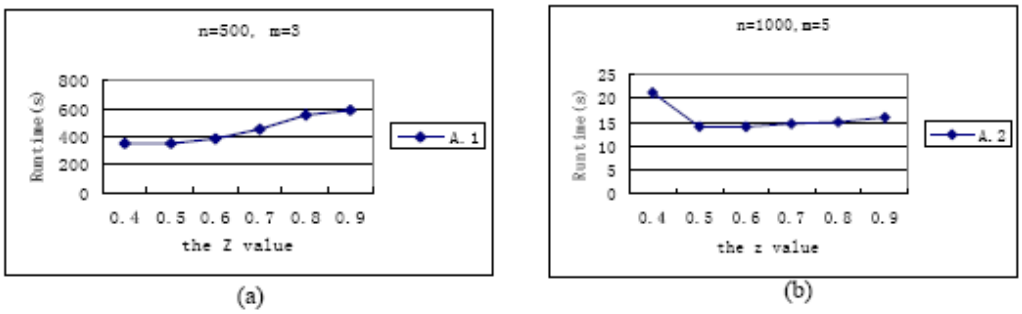


Fig. 4. Characters of fast re-generalization for the two algorithms

6.2 Application in a Real Dataset

A plant distributing dataset, which involves 29 plant species which are very valuable and rare in "The Three Parallel Rivers in Yunnan Area" zone and 319 instances (tuples), is used in our experiments. Table 1 is some tuples of this dataset.

Table 1. Some tuples of a plant distributing dataset

Tuple-ID	Plant-name	Veg-name	Elevation /m	Location
t_1	Orchid	meadow	[1000, 1500]	Lijiang
t_2	Fig	scrub	[2400, 3000]	Weixi
t_3	Magnolia	scrub	[3000, 3700]	Lijiang
t_4	Calligonum	taiga	[2000, 3000]	Jianchuan
t_5	Magnolia	meadow	[3000, 4000]	Lanping
t_6	Agave	taiga	[3000, 4000]	Lanping
t_7	Yucca	forest	[1500, 2400]	Weixi
t_8	Waterlily	meadow	[800, 2200]	Jianchuan

The experiments using this dataset are aimed at checking the usefulness of the AOG-ags algorithms. Can they discover valuable patterns? Are the rules discovered by our algorithms interesting towards geographers and botanists?

34 AGS are obtained when the threshold Z is set 0.8, and 57 plant distributing rules are discovered when one of the 34 AGS is chosen according to their interestingness. When the threshold Z is set 0.85, the number of AGS is 28 and the number of rules is 19. When the Z is set 0.9, the number of AGS and rules is 22 and 16 respectively.

Some rules discovered by our algorithms are really attractive to geographers and botanists. The followings are some examples:

- "Tricholoma matsutake" ⇒ 40% grows in the forest and meadow whose elevation is from 3300 to 4100 meter of Lijiang.
- "Angiospermae" ⇒ 80% grows in the forest、scrub and meadow whose elevation is from 2400 to 3900 meter of Lijiang and Weixi.
- Lijiang ⇒ There are a plenty of plants species in severe danger such as "Tricholoma matsutake", "Angiospermae", "Gymnospermae", and so on.

7 Conclusions

Related approaches for mining the associations of attributes can be divided into the clustering-based approach, the association rule-based method and the approach of AOG. Clustering-based approach treats every attribute as a layer and considers clusters of point-data in each layer as candidates for mining associations [7, 8]. The complexity and the low-efficiency are the crucial problems of this method. The association rule-based approach is divided into the transaction-based method and distance-based method again. The transaction-based method computes the mining transaction (two-dimension table) by a reference-object centric model, so one can use the method

which is similar to *Apriori* for mining the rules [9,10]. The problem of this method is that a suitable reference-object is required to be specified. The distance-based method was proposed by Morimoto [11], and Shekhar together with Huang [12-14] did further research. Because of doing a plenty of join operations, executing efficiency is the key problem of this method. The approach of AOG is presented firstly by Cai [1]. It is a simple and understandable method. But it is inconvenient because setting each attribute threshold is required.

The AOG-ags proposed in this paper can obtain automatically rules under setting a threshold Z. Particularly, by using the AOG-ags algorithm in a plant distributing dataset, some distributing rules for the species of plants in "Three Parallel Rivers of Yunnan Protected Areas" zone are found interesting. The advantage of AOG method is that expert knowledge (concept hierarchy trees) is used in the process of data mining.

References

1. Y. Cai, N. Cercone, and J. Han. Attribute-Oriented Induction in Relational Databases, G. Piatetsky-Shapiro and W.J. Frawley, eds., Knowledge Discovery in Databases, AAAI/MIT Press, Menlo Park, Calif., pp. 213-228, 1991.
2. J. Han. Towards Efficient induction mechanisms in database systems. Theoretical Computering Science, 1994, 133:361-385.
3. L. Wang. A method of the abstract generalization on the bases of the semantic proximity. CHINESE J. COMPUTERS, Vol.23 No. 10, pages 1114-1121, Oct. 2000.
4. C. L. Carter, H. J. Hamilton. Efficient attributed-oriented generalization for knowledge discovery from large databases. IEEE Trans. on Knowledge and Data Eng., 1998, 10(2):193-208.
5. H. Chen, L. Wang. Quantifiable Attribute-Oriented Generalization. Journal of Computer Research & Development, Vol. 38, No. 2, pages 150-156. Feb. 2001.
6. J. Han and M. Kamber, Data mining: concepts and techniques, Morgan Kaufmann publishers, San Fransisco, 2001.
7. V. Estivill-Castro and I. Lee. Data Mining Techniques for Autonomous Exploration of Large Volumes of Geo-Referenced Crime Data, Proc. Sixth Int'l Conf. Geocomputation, 2001.
8. V. Estivill-Castro and A. Murray. Discovering Associations in Spatial Data—An Efficient Medoid Based Approach, Proc. Second Pacific-Asia Conf. Knowledge Discovery and Data Mining, 1998.
9. K. Koperski and J. Han. Discovery of Spatial Association Rules in Geographic Information Databases. Proc. Fourth Int'l Symp. Spatial Databases, 1995, pp. 47-66.
10. L. Wang, K. Xie, T. Chen and X. Ma. Efficient discovery of multilevel spatial association rule using partition. Information and Software Technology (IST). Volume 47, Issue 13, 2005, P. 829-840.
11. Y. Morimoto. Mining Frequent Neighboring Class Sets in Spatial Databases. Proc. ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining, 2001. pp. 353-358.
12. H. Xiong, S. Shekhar, Y. Huang, V. Kumar, X. Ma, and J. S. Yoo. A Framework for Discovering Co-location Patterns in Data Sets with Extended Spatial Objects. In Proc. 2004 SIAM International Conference on Data Mining (SDM), 2004, 1-12.
13. Y. Huang, S. Shekhar and H. Xiong. Discovering Colocation Patterns from Spatial Data Sets: A General Approach. IEEE Transactions on Knowledge and Data Engineering, 16(12) 2004, pp. 1472-1485,
14. J. S. Yoo and S. Shekhar. A partial Join Approach for Mining Co-location Patterns, In Proc. of the 12th annual ACM international workshop on Geographic information systems, 2004, 241-249.

An Effective Approach to Predicting Plant Species in an Ecological Environment*

Lizhen Wang[1]
Department of Computer Science and Engineering,
School of Information Science and Engineering
Yunnan University
Kunming, 650091, China

Joan Lu, Jim Yip
Department of Informatics, School of Computing and Engineering
University of Huddersfield
Huddersfield, UK, HD1 3DH

Abstract - *Using data mining method in the ecosystem modeling, the correlation between a plant and certain ecological environment can be discovered. This is helpful for the reconnaissance and the exploitation of rare plants. Based on the semantic proximity, a mining method to evaluate the fuzzy association degree is given in this paper. Inverse document frequency (IDF) weight function has been adopted in this investigation to measure the weights of ecological environments in order to superpose the fuzzy association degrees. To implement the method, the "growing window" and the proximity computation pruning are introduced to reduce both I/O and CPU costs in computing the fuzzy semantic proximity between time-series. Extensive experiments on real datasets are conducted, and the results show that the mining approach is reasonable and effective.*

Keywords: Data mining; Fuzzy association degree; Time-series data; Fuzzy equivalence partition; Three-Parallel-River zone.

1 Introduction

An important task in data mining is to mine the correlation among entities. Data mining is suitable for exploring large dataset for patterns and systematic relationships between variables [3, 4].

In this paper, data mining is employed to explore a plant species and ecological environment dataset compiled from plant species data, climate data (mean temperature, mean precipitation), elevation data, and topography data (e.g., peak, valley, ascent, terrace, and basin) (see Table 1). The ultimate goal of this investigation is to discover the association pattern between plants and ecological environments, and then to predict certain plant species in an ecological environment.

There are number of methods in the mining prediction, such as statistical learning method [10], machine learning [11], decision tree method [12], and fuzzy method [13]. Further more, Carter [7, 8] handles data generalization by the attribute-oriented generalization method. Wang [5] proposes a fuzzy equivalent partition method to handle data generalization. A data reduction technique based on attribute-oriented generalization is presented in paper [6]. Shi [9] presents a novel data preprocessing technique called shrinking inspired by the Newton's Universal Law of Gravitation in the real world, which optimizes the inner structure of data.

In this research, based on the semantic proximity expression, the fuzzy association degree with superposition approach can be evaluated. It is shown that these approaches are reasonable and effective.

2 Methods employed

2.1 Dataset preparation

It is believed that the ecological environments, including climates (e.g., mean temperature, mean precipitation), elevation, soil, topography, etc, contribute to the distribution and diversity of plant species in Three-Parallel-River zone [1, 2]. Discovering correlations between the plant species and ecological environments is significant for retaining rare and endangered plants. The data listed in Table 1 will be used in the research. The attributes are relevant to the problem stated here.

* Supported by the National Natural Science Foundation of China under Grant No.60463004.

[1] PhD student in Department of Informatics, School of Computing and Engineering, University of Huddersfield, Huddersfield, UK, HD1 3DH.

Table 1 Plant species and ecological environment dataset

Tuple-ID	mean temperature (monthly 0.1°C)	mean precipitation (monthly 0.1mm)	elevation (m)	topography	plant species
r_1	90, 100, 108, 130, …	0, 7, 21, 21, 307, …	[900,2000]	ascent	Camellia
r_2	80, 111, 130, 102, …	12, 13, 133, 55, …	[500,900]	ascent	Water-lily
r_3	99, 100, 144, 142, …	71, 205, 502, 330, …	[700,1100]	valley	Camellia
r_4	93, 115, 141, 165, …	0, 98, 171, 793, …	[200,700]	ascent	Camellia
r_5	77, 68, 116, 113, …	17, 228, 212, 453, …	[120,400]	valley	Water-lily
r_6	93, 105, 130, 145, …	36, 228, 679, 190, …	[200,800]	valley	Orchid
r_7	93, 103, 120, 151, …	40, 882, 46, 899, …	[600,1200]	basin	Orchid
r_8	67, 84, 81, 105, …	7, 62, 68, 184, 734, …	[1000,2000]	ascent	Water-lily

2.2 Predicting processes

There are two processes involved in the investigation: 1) the initial data exploration and 2) the mining prediction. The initial data exploration process converts the original data into mining data for prediction. The mining prediction performs the predicting results.

2.2.1 Initial data exploration - IDE

The Initial Data Exploration (IDE) usually starts with data preparation that may involve data cleaning, data transformation, and data generalizations that can be applied to obtain a reduced representation of the data set that is much smaller in volume, yet closely maintains the integrity of the original data. Prior to IDE, data analysis is carried out as follows.

1) Climate data: there are two groups of time-series data for mean temperature and mean precipitation. These data change over time.

2) Elevation data: it is the interval values.

3) Topography data: it may have the following values: peak, valley, ascent, terrace, and basin…, all of them are used to describe the location of the plants growth.

In the light of above, the key following issues are going to tackle in the study:

- Comparing the similarity between two time-series
- Approximately partitioning them

a. Comparing the similarity of two time-series

A time-series is a sequence of real numbers representing values at specific points in time. We start by defining time-series.

Definition 1. A time-series T of length n is an ordered set $(t_1, t_2, \ldots, t_n)$ with $t_i \in \Re$, $1 \le i \le n$. |T| is the length of T.

Definition 2. Given two time-series, T[1…n] and Q[1…n], the degree of proximity between T and Q (denoted PD (T, Q), $0 \le PD(T,Q) \le 1$) is defined as

$$PD(T,Q) = 1/\sqrt{\sum_{i=1}^{n}(t_i - q_i)^2 + 1} \qquad (1)$$

Simple, yet important, this property held by the PD is described by the lemma 1. This property will be useful in improving the performance of computing PD. Simply put the lemma 1 states that if two time-series are in close proximity, then all their prefix subsequences of equal length are also in close proximity.

Lemma 1. If $PD(T[1...n], Q[1...n]) > \varepsilon$ for time-series T[1…n] and Q[1…n] then for any $1 \le k \le n$, $PD(T[1...k], Q[1...k]) > \varepsilon$ holds.

Proof. (By contradiction) If for a particular k, $1 \le k \le n$, $PD(T[1...k], Q[1...k]) \le \varepsilon$, however, $PD(T[1...n], Q[1...n]) \le PD(T[1...k], Q[1...k])$, and therefore $PD(T[1...n], Q[1...n]) \le \varepsilon$, a contradiction, since we assumed that $PD(T[1...n], Q[1...n]) > \varepsilon$. □

Using a "growing window" to scan the time-series, the computation of PD(T, Q) can be done recursively by adding the remaining terms to the previously sums, thus the number of needed computations are reduced. For example, if the values: PD(T[1…300], Q[1…300]) have been computed, then the values of PD(T[1…301], Q[1…301]) can be computed directly using Equation (2).

$$PD(T[1...301], Q[1...301]) = 1/\sqrt{\frac{1}{(PD(T[1...300], Q[1...300]))^2} + (T[301] - Q[301])^2} \qquad (2)$$

This allows us to perform an efficient "growing window" algorithm. For example, if we compute the arguments in (1) for a window of size m in T and Q, i.e., PD(T[1…m], Q[1…m]), we can compute the same arguments for the "growing" window PD(T[1…m+1], Q[1…m+1]) in O(1) time.

Using the "growing window" and Lemma 1, there is an efficient pruning strategy. When a value of PD from a

growing window is less than ε, it can be considered as zero, and no further computation is needed.

b. Fuzzy equivalence partition for the set of time-series

Once the value of PD between two time-series obtained, the remained issue is how to partition the set of time-series. Fuzzy equivalence partition method [5, 14, 15] is used as follows.

Assume that $T_1,\cdots,T_N$ is a set of time-series. From the Definition 2, there is a relationship $PD(T_i,T_i)=1$ and $PD(T_i,T_j)=PD(T_j,T_i)$. Using the degree of proximity between two time-series, a similarity matrix $S=(s_{ij})_{N\times N}$ can be built up, where

$$s_{ij}=\begin{cases}1 & i=j\\ PD(T_i,T_j) & i\neq j\end{cases} \qquad (3)$$

S can be multiplied by itself repeatedly, where $(s_{ij})^2=MAX_k(MIN(s_{ik},s_{kj}))$, until $S^{2k}=S^k$. S^{2k} is called, which is a fuzzy equivalence matrix EM. Based on the fuzzy equivalence matrix EM, the classifications of $T_1,\cdots,T_N$ will be obtained for user-specified level value ρ.

The equivalence matrix EM can be computed in $O(N^3)$ time from a similarity matrix S. The computational method can be expressed as follows:

```
1 For i:=1 to N do
2   For j:=1 to N do
3     If s(j,i)>0 then
4       For k:=1 to N do
5         S(j,k):=max{S(j,k), min{S(j,i),
S(i,k)}};
```

This algorithm is very efficient if the fuzzy similarity matrix S has many zero elements, due to step 3.

The above method can also be optimized through computing the fuzzy equivalence matrix EM in $O(N^2)$ time (the exact time is $O(M^2)$, $O(M^2) < O(N^2)$, M is the number of nonzero elements in the upper-triangle of the similarity matrix S.) [5].

2.2.2 Mining prediction

a. Fuzzy association degree between plants and ecological environments

Let $R=\{r_1, \ldots, r_p\}$ be a finite set of objects, $A=\{A_1, \ldots, A_n, B\}$ be a set of attributes over R. The attributes in A are classified into disjoin conditional attributes $C=\{A_1, \ldots, A_n\}$ and the decision attribute $D=\{B\}$. The equivalence class L in the set of the equivalence classes for the conditional attributes A_k is denoted by A_{kL}, and for decision attribute B, the B_m means the m^{th} equivalence class. The intersection of A_{kL} and B_m is denoted by $POS(A_{kL}, B_m)$. The number of objects in $POS(A_{kL}, B_m)$ is called the distribution of A_{kL} to B_m, written as $K(POS(A_{kL}, B_m))$.

The degree of fuzzy association between A_{kL} and B_m is determined as follows. Suppose A_{kL} represents a monthly mean temperature time-series data, and B_m shows the type of plant. Once the mean temperature is A_{kL}, the plant species must be B_m, then A_{kL} is in close association with B_m. In another condition, the mean temperature is A_{kL}, and the plant species is B_m or B_h, then the fuzzy association degree between A_{kL} and B_m can be decided depending on the proximity between B_m and B_h. The greater the proximity between B_m and B_h, the higher the fuzzy association degree between A_{kL} and B_m. If the plant species are B_m, B_h, or B_f when the mean temperature has the value A_{kL}, it is unlikely to have a relationship between A_{kL} and B_m.

Definition 3 Given a weight w_{jm} for measuring the proximity between equivalence class B_j and equivalence class B_m in decision attribute B, the fuzzy association degree from A_{kj} to B_m is defined as $(0 \le \lambda \le 1)$

$$\begin{aligned}\lambda(A_{kj},B_m)=&(K(POS(A_{kj},B_1))*w_{1m}\\ &+K(POS(A_{kj},B_2))*w_{2m}\\ &+\cdots+K(POS(A_{kj},B_s))*w_{sm})/K(A_{kj})\end{aligned} \qquad (4)$$

The weight w_{jm}, written as $w(B_j, B_m)$, represents the degree of proximity between B_j and B_m. $w(B_m, B_m)=1$ and $w(B_j, B_m)=w(B_m, B_j)$ are held. The weight w_{jm} can be obtained from botanists, or by mining the co-location relation between plants [16-18]. The following method is adopted in this study.

Plants can be organized to a plant family tree as its leaf nodes describe an association relationship of plants. It is called as the conceptual hierarchy tree.

Definition 4 Given a conceptual hierarchy tree, f_1 and f_2 are two leaves of this tree, the value of weight $w(f_1, f_2)$ is defined as:

$$w(f_1,f_2)=\begin{cases}1 & f_2=f_1\\ 1-l(P(f_1,f_2))/H & f_2\neq f_1\end{cases} \qquad (5)$$

Where, $P(f_1,f_2)$ returns the node of the common parent, $l(f)$ gets the level number of node f and H is the height of the tree.

The weight w satisfies the following properties:

(1) For any plant f in a tree, $w(f,f)=1$ holds.

(2) For the parent of two plants f_1 and f_2 is the root in a tree, $w(f_1,f_2)=0$ holds.

(3) For any two plants f_1 and f_2 in a tree, $0 \le w(f_1, f_2) \le 1$.

(4) For any plants f_1, f_2, f_3 and f_4 in a tree, if $l(P(f_1f_2)) > l(P(f_3, f_4))$, then $w(f_1f_2) < w(f_3f_4)$.

b. Superposition of the degrees of fuzzy association

The factors affecting the plant species B_m are not only the values A_{1k} of conditional attribute A_1, but also A_{2f}, A_{3h},... which are superposed as follows:

Definition 5 The superposition of $\lambda_1, \lambda_2, \ldots, \lambda_k$, written as $\lambda_1 \oplus \lambda_2 \oplus \cdots \oplus \lambda_k$, is defined as

$$\lambda_1 \oplus \lambda_2 \oplus \cdots \oplus \lambda_k = [\sum_{i=1}^{k} \mu_i \lambda_i^{\,2}]^{1/2} \qquad (6)$$

Where μ_i is the weight on λ_i.

Inverse document frequency (IDF) weight function is adapted to the relational domain by treating each tuple as document of attribute values. The weight of attribute value 'a_1' is expected to be less than that of 'd_3', since 'a_1' appears more frequently as a value for its attribute than 'd_3' does.

Let the frequency of attribute value 'a' in the attribute A_i, denoted freq(a, A_i), be the number of tuples (i.e., objects) r in R such that $r[A_i] =$ 'a'. The *IDF* value, $IDF(a, A_i)$, of an attribute value 'a' with respect to the attribute A_i in the schema of R is computed as Equation (7), when $freq(a, A_i) > 0$,

$$\mu(a, A_i) = IDF(a, A_i) = \log \frac{K(T)}{freq(a, A_i)} \qquad (7)$$

For an attribute value 'a' whose frequency in attribute A_i is 0, 'a' is an erroneous version of some values in the reference tuple. Since we do not know the value to which it corresponds, the weight $\mu(a, A_i)$ is defined to be the average weight of all values in this attribute of relation R.

c. An example

Example 1 Suppose Table 2 is the result after preprocessing the data of Table 1, where A_1, A_2, A_3, A_4 is the attribute *mean temperature*, *mean precipitation*, *elevation*, and *topography* respectively. Attribute B is the *plant species*. We use the same lower case letter if $r_i[A_k]$, $r_j[A_k]$ belong to the same equivalence class (For the attribute *elevation* (A_3), it can be partitioned based on the concept of the semantic proximity between two interval values. In this study, the semantic proximity between two interval values f_1 and f_2 can be defined as $SP(f_1, f_2) = (size(f_1 \cap f_2) / size(f_1 \cup f_2))$, where $size(h)$ i the size of interval h).

There is a new ecological environment data in Table 3 (i has been preprocessed). Let us predict the species of plan under this ecological environment. We obtain th distributing table from 'a_2', 'b_2', ... to f_1, f_2 or f_3 (see Tabl 5) and corresponding values of w (see Table 4), where fo f_1 we have w_{11}= 1, w_{12}= 0, w_{13}=0, etc. In the same way, w compute the $\mu(a, A_i)$. We obtain Table 6.

Table 2 An example of data table for prediction

Tuple-ID	A_1	A_2	A_3	A_4	B
r_1	a_1	b_1	c_1	e_1	f_1
r_2	a_2	b_1	c_2	e_1	f_2
r_3	a_1	b_2	c_1	e_2	f_1
r_4	a_3	b_3	c_2	e_1	f_1
r_5	a_2	b_2	c_2	e_2	f_2
r_6	a_2	b_4	c_2	e_2	f_3
r_7	a_1	b_4	c_1	e_3	f_3
r_8	a_3	b_3	c_1	e_1	f_2

Table 3 A new ecological environment data that have been preprocessed

A new ecological environment data	A_1	A_2	A_3	A_4
r	a_2	b_2	c_1	e_1

Table 4 Weight w for the decision attribute B

w	f_1 (Camellia)	f_2 (Water-lily)	f_3 (Orchid)
f_1	1(w_{11})	0(w_{12})	0(w_{13})
f_2	0(w_{21})	1(w_{22})	0.6(w_{23})
f_3	0 (w_{31})	0.6 (w_{32})	1(w_{33})

Table 5 The distributing table

K	f_1	f_2	f_3
a_2	0	2	1
b_2	1	1	0
c_1	2	1	1
e_1	2	2	0

Table 6 Weight μ for superposition

μ	a_2	b_2	c_1	e_1
	0.4260	0.6021	0.3010	0.3010

Thus we have

$$\lambda(a_2, f_1) = [K(Pos(a_2, f_1)) \times w_{11} + K(Pos(a_2, f_2)) \times w_{12} + K(Pos(a_2, f_3)) \times w_{13}] / 3 = 0*1+2*0+1*0=0$$

$$\lambda(a_2, f_2) = [K(Pos(a_2, f_1)) \times w_{21} + K(Pos(a_2, f_2)) \times w_{22} + K(Pos(a_2, f_3)) \times w_{23}] / 3 = (0*0+2*1+1*0.6)/3=2.6/3=0.867$$

Using the same method, we obtain Table 7. Superposing the data in row f_1, we have the result of association degree. It indicates the association degree of the ecological environment data with 'a_2', 'b_2', 'c_1', and 'e_1' to plant species 'f_1', 'f_2', or 'f_3'. It means we can predict that the plant f_2 (Water-lily) might grow in the ecological environment with 'a_2', 'b_2', 'c_1', and 'e_1'.

Table 7 Mining prediction of plant species

λ	a_2	b_2	c_1	e_1	The result of superposition
f_1	0	0.5	0.5	0.5	0.5487
f_2	0.867	0.5	0.4	0.5	0.7708
f_3	0.733	0.3	0.4	0.3	0.5986

3 Results and discussion

3.1 Estimating error rates

The bootstrapping method was performed to estimate the error rates. In experiments, 34 kinds of plant species were selected from the plant dataset. The species of plants have been predicted according to the fuzzy association degree threshold. If the results of the algorithm are less than the threshold, the ecological environment would be refused to be predicted. False positives (FP%) and false negatives (FN%) are used to evaluate the algorithm's error rates.

Table 8 shows the results of FP% and FN% with given the different thresholds of the fuzzy association degree for the 34 plant species. In the experiments, we fixed the equivalence partition level value thresholds ρ_1=0.000224, ρ_2=0.0045, for two time-series attributes respectively. It has been observed that to lower level the thresholds of fuzzy association degree will increase the percentage of false positive.

Table 8 FP% and FN% for 34 plants

Thresholds of fuzzy Association degree / Error rates	0.6	0.5	0.4	0.3	0.2
FP%	21.4	26.7	26.7	26.5	26.5
FN%	50	75	75	null	null

3.2 Quality

Beside the error rate, other parameters have been used to measure the prediction performance, i.e. *Sensitivity=t_pos/pos, Specificity=t_neg/neg, Precision=t_pos/(t_pos+t_neg), and Accuracy=Sensitivity*(pos/(pos+neg))+Specificity*(neg/(pos+neg))*

Table 9 shows all results with using different measures for both 15 plants and 34 plants. In the experiments, the equivalence partition level value thresholds ρ_1=0.000224, ρ_2=0.0045, have been fixed and the threshold of the fuzzy association degree is 0.4.

Table 9 Results of the algorithm's quality

Quality / Plants	Sensitivity	Specificity	Precision	Accuracy
15 plants	9/12=75	1/3=33.3	9/10=90	66.7
34 plants	22/30=73.3	1/4=25	22/23=95.7	67.6

3.3 Performance of the algorithm

The program was run on artificial datasets of size 15, 30, 50, …, 1000 plants. Fig. 1 shows that this program is increasing quickly with the increasing size of datasets, because the large time-series data need to be dealt with for just adding a tuple into the dataset.

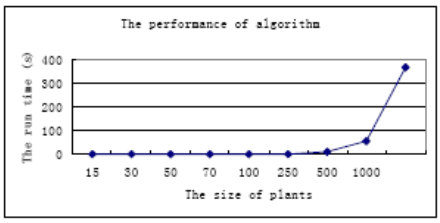


Figure 1 The performance of algorithm

4 Conclusions and future work

This research makes contribution to using data mining techniques to resolve the problems in predicting plant species in an ecological environment in Three-Parallel-River zone. It has been found that using the fuzzy association degree with superposition approach could achieve reasonable and effective results.

For future works, it will be carried out formally to characterize the relative strengths and weaknesses of various prediction tests and to study the confidence of prediction results. Other interesting directions are mining a functional dependency relationship between condition attributes, and mining crucial factors which affect plant

growing, which could be advantageous to protect and retain rare and endangered plants.

Acknowledgments The authors would like to thank student Ao Yang for helping us to implement the experiments. This research is supported by the National Natural Science Foundation of China (No. 60463004)

5 References

[1] Liqun Guo. Geological Dividing of Forests at Three Parallel Rivers of Yunnan Protected Areas, Journal of West China Forestry Science, 33 (2) (2004) 10-15

[2] Mingchun Peng, Shuhua Yang, and Haizhong Yang. The Study on Design of Yunnan Information System for Biodiversity Conservation (YBCIS), Journal of Yunnan University, 21 (2) (1999) 90~93

[3] J. Han and M. Kamber, Data mining: concepts and techniques, Morgan Kaufmann publishers, San Francisco (2001)

[4] H. A. Edelstein. Introduction to data mining and knowledge discovery (3rd ed). Potomac, MD: Two Crows Corp (1999)

[5] L. Wang. A method of the abstract generalization on the bases of the semantic proximity. CHINESE J. COMPUTERS, Vol.23 No. 10, (2000) 1114-1121

[6] L. Wang and H. Chen. Record Reduction Based on Attribute Oriented Generalization, ICMLC2005, Proceedings of the Fourth International Conference on Machine Learning and Cybernetics (ICMLC), Guangzhou, China (2005) 1693-1700

[7] C.L.Carter and H.J.Hamilton, Fast, Incremental Generalization and Regeneralization for Knowledge Discovery from Databases, Proc. FLAIRS, Eighth Florida Artificial Intelligence Research Symp., Melbourne, Fla. (1995) 319-323

[8] Colin L.Carter and Howard J.hamilton, Efficient Attribute-Oriented Generalization for Knowledge Discovery from Large Databases, IEEE Trans. On Knowledge and Data Engineering, Vol.10, No.2, March/April (1998) 193-208

[9] Y. Shi, Y. Song and A. Zhang, A Shrinking-Based Approach for Multi-Dimensional Data Analysis, proceedings of the 29th International Conference on Very Large Data Bases (VLDB03), Berlin, Germany (2003)

[10] T. Hastie, R. Tibshirani, and J. H. Friedman. The Elements of Statistical Learning: Data mining, Inference, Prediction. Springer, New York, 2001.

[11] I. H. Witten and E. Frank. Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann, 1999.

[12] K. Alsabti, S. Ranka, and V. Singh. CLOUDS: A Decision Tree Classifier for Large Datasets. In Proc. of the 4th Intl. Conf. on knowledge Discovery and Data Mining, p. 2-8, New York, August 1998.

[13] W. Liu, N. Song. The fussy association degree in semantic data models. Fuzzy Sets and Systems 117(2001) 203-208.

[14] Zhong-Xiong Huo, Fuzzy Mathematics and its Applications, Tianjin Science and Technology Press, Tianjin, China (1989)

[15] L.A. Zadeh, Fuzzy sets, Inform. and Control 8 (3) (1965) 338-353

[16] Y. Huang, S. Shekhar and H. Xiong. Discovering Colocation Patterns from Spatial Data Sets: A General Approach. IEEE Transactions on Knowledge and Data Engineering, 16(12) (2004) 1472-1485

[17] J. S. Yoo and S. Shekhar, A partial Join Approach for Mining Co-location Patterns, In Proc. of the 12th annual ACM international workshop on Geographic information systems (2004) 241-249

[18] H. Xiong, S. Shekhar, Y. Huang, V. Kumar, X. Ma, and J. S. Yoo. A Framework for Discovering Co-location Patterns in Data Sets with Extended Spatial Objects. In Proc. 2004 SIAM International Conference on Data Mining (SDM) (2004) 1-12.

A New Join-less Approach for Co-location Pattern Mining*

Lizhen Wang [a, b], Yuzhen Bao [a], Joan Lu [b], Jim Yip [b]
[a] *Department of Computer Science and Engineering, School of Information, Yunnan University, Kunming, 650091, P. R. China*
[b] *Department of Informatics, School of Computing and Engineering, University of Huddersfield, Huddersfield, UK, HD1 3DH*
lzhwang@ynu.edu.cn

Abstract

With the rapid growth and extensive applications of the spatial dataset, it's getting more important to solve how to find spatial knowledge automatically from spatial datasets. Spatial co-location patterns represent the subsets of features whose instances are frequently located together in geographic space. It's difficult to discovery co-location patterns because of the huge amount of data brought by the instances of spatial features. A large fraction of the computation time is devoted to generating the table instances of co-location patterns. The essence of co-location patterns discovery and three kinds of co-location patterns mining algorithms proposed in recent years are analyzed, and a new join-less approach for co-location patterns mining, which based on a data structure---- CPI-tree (Co-location Pattern Instance Tree), is proposed. The CPI-tree materializes spatial neighbor relationships. All co-location table instances can be generated quickly with a CPI-tree. This paper proves the correctness and completeness of the new approach. Finally, an experimental evaluation using synthetic datasets and a real world dataset shows that the algorithm is computationally more efficient than the join-less algorithm.

1. Introduction

Spatial data mining is the process to discover interesting and previous unknown, but potential useful patterns from spatial datasets [1,2]. Extracting interesting patterns from spatial datasets is more difficult than extracting the corresponding patterns from transaction datasets due to the complexity of spatial data types, spatial relationships and spatial autocorrelation [3]. A spatial co-location pattern represents a subset of spatial features whose instances are frequently located in a spatial neighborhood. Spatial co-location patterns may yield important insights for many applications. For example, a mobile service provider may be interested in mobile service patterns frequently requested by geographical neighboring users. The locations which are gotten together by people can be used for providing attractive location-sensitive advertisements, etc. Other application domains include Earth science, public health, biology, transportation, etc.

In previous work on mining co-location patterns, Morimoto[8] defined distance-based patterns called k-neighboring class sets. In his work, the number of instances for each pattern is used as the prevalence measure, which does not possess an anti-monotone property by nature. However, Morimoto used a non overlapping instance constraint to get the anti-monotone property for this measure. In contrast, Shekhar & Huang [7] developed an event centric model, which does away with the non-overlapping instance constraint. The related works in the approach proposed by Shekhar & Huang can be classified into three kinds for identifying co-location table instances: the full-join approach [6], the partial-join approach [9] and the join-less approach [10].

The *full-join* approach is mainly based on the computation of the join operation between table instances for identifying co-location instances. This approach is similar to *Apriori* method and it could generate correct and complete prevalent co-location sets. However, scaling the algorithm to substantially large dense spatial datasets is challenging due to the increasing number of co-locations and their table instances.

The *partial-join* algorithm proposed by Yoo and Shekhar in [9] is to build a set of disjoint clique in

* Supported by the National Natural Science Foundation of China under Grant No.60463004.

spatial instances to identify the intraX instances of co-location (belonging to a clique) and interX instances of co-location (belonging to between two cliques), and join the intraX instances and interX instances respectively to calculate the value of PI (The participation index). The key of this algorithm is to find out cliques as big as possible, which could cut down the spatial neighbor relationships between two cliques. Besides building cliques is time-consuming, if the correct cliques could not be identified, and the number of cut neighbor relations would not be decreased, the partial-join algorithm of mining co-location pattern would be similar to the full-join algorithm.

The *join-less* algorithm proposed by Yoo, Shekhar and Celik in [10] puts the spatial neighbor relationships between instances into a compressed *star neighborhood*. All the possible table instances for every co-location were generated by scanning the star neighborhood, and by 3-time filtering operations. The join-less approach is efficient since it uses an instance-lookup scheme instead of an expensive spatial or instance join operation for identifying table instances. However, the star neighborhood structure is not an ideal structure for generating table instances, for the table instances generating from this structure have to be filtered. Therefore, the computation time of generating table instances will increase with the growing of length of co-location patterns.

In the paper, a new structure called *CPI-tree* (Co-location Pattern Instance Tree) is introduced. It could materialize the neighbor relationships of a spatial data set, and find all the table instances recursively from it. Different from the star neighborhood structure in the join-less approach, all information of the neighbor relationships in a spatial dataset is organized together by the CPI-tree. So, the third phase filters in the join-less algorithm, which might be an expensive step, need not be performed. Meanwhile, some filtering methods appeared in the join-less algorithm, which can filter candidate co-locations without finding exact co-location instances, will be reserved. Some new filtering methods will be considered in our new algorithm. Although, in many cases the *Apriori* candidate generate-test method reduces the size of candidate sets significantly and leads to performance gain. However, it may need to repeatedly scan the star neighborhood and check a large set of candidates by pattern matching. This is especially the case for mining long patterns.

The reminder of the paper is organized as follows. The CPI-tree structure and its construction method are introduced in Section 2. Section 3 develops a CPI-tree-based complete table instances generating algorithm. The experimental results are presented in Section 4. Section 5 summarizes our study and points out some future research issues.

2. Colocation-pattern Tree (CPI-tree): Design and Construction

2.1. CPI-tree

A compact data structure can be designed based on the following observations:

(1). Since spatial neighbor relationships between instances make certain all table instances, it is necessary to perform one scan of spatial datasets to identify the set of spatial neighbor relationships.

(2). If the set of neighbor relationships can be stored in a compact data structure, it may be possible to avoid repeatedly scanning the set of neighbor relationships. (*apriori-like* algorithms did so, because they have to scan all size-k table instances when they generating size-k+1 table instances.)

(3). If a size-k table instance is found, it may be cost-efficient to expanding a size-k+1 table instance from it at once. It is easy to expand table instance if the set of neighbor relationships can be stored in a compact data structure.

(4). The recursive and hierarchical properties of tree structure makes the clearness and simpleness of describing algorithms based on the tree structure. If all spatial instances are sorted in ascending order (the spatial features in alphabetic order, and then the different instance of the same spatial feature in numerical order), a graph G representing spatial neighbor relationships may correspond to a unique tree structure.

With the above observations, a tree structure (called CPI-tree (Co-location Pattern Instance Tree), for all table instances can be generated from it) can be defined as follows.

Definition 1 (CPI-tree). A CPI-tree is a kind of rooted tree. The root of CPI-tree is labeled as "null". A branch of the CPI-tree is constructed a corresponding connective sub-graph in the graph G. The node in CPI-tree represents the spatial instance. The node u is the parent of the node v, when there is a neighbor relationship between instances u and v and the instance u is "smaller" than instance v.

Based on this definition, there is the following CPI-tree construction approach.

1) Create the root of a CPI-tree, and label it as "null".

2) Push all the spatial instances into a stack T1 in alphabetic and numerical descending order.

3) Pop an instance from the stack T1, create a child node of the root "null" for this instance (e.g. A.1 in fig.2). Push this instance into a stack T2.

4) Pop an instance (e.g. A.1) from stack T2. Find out all the instances which are the neighbors of this instance (showed in Fig. 1), the different spatial features from this instance, and "bigger" than this instance. These instances form child nodes of this node in ascending order. Delete all these instances from stack T1 and stack T2, and link to the same instance-name node (except for leaf-nodes) in the CPI-tree (see dashed link in Fig.2).

5) Push child node instances into the stack T2 in descending order. Then, turn to 4).

6) Repeat the operation above till the stack T2 is empty, and then turn to 3).

7) Repeat the operation above till the stack T1 is empty.

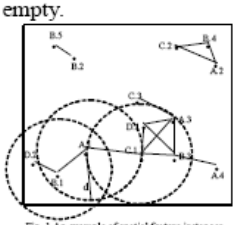


Fig. 1 An example of spatial feature instances and neighbors of the instance A.1, B.1 and C.1 (connecting them with a solid line when a pair of spatial instances satisfies spatial neighbor relationship)

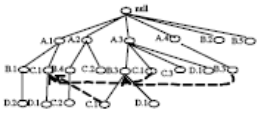


Fig. 2 CPI-tree of the example in fig.1

The Fig. 2 is the CPI-tree of the example in fig.1. The CPI-tree of a spatial dataset constructed by above steps will be unique. The CPI-tree materializes the neighbor relationships of a spatial dataset with no duplication of the neighbor relationships and no loss of co-location instances, and more important thing is that it is convenient and efficient to generate the co-location instances from it.

The approach of constructing a CPI-tree can be converted into the following algorithm.

Algorithm 1 (CPI-tree construction).

Input: *S*: a set of spatial instances and each instance is a vector <feature type, instance id, location>; *R*: the spatial neighbor relationship (e.g. Euclidean distance);

Variables: $NT=\{NT_{l_1}, NT_{l_2}...NT_{l_m}\}$: a set of spatial neighbor relationships where NT_{l_i} is the set of neighbor instances of the instance l_i, whose order is "bigger" than l_i and is sorted in ascending order.

Output: *CPI-tree*: the CPI-tree structure of materialized spatial neighbor relationships;

Method:

1) *NT=gen_neighborhood;*
2) Create the root of a CPI-Tree, and label it as "null."
3) Push all the instances in *S* into a stack T1 in descending order;
4) **While** the stack T1 is not empty **Do**
5) { *l* ← Pop an instance from stack T1;
6) create the node *l* which is the child of the root "null";
7) Push instance *l* into the stack T2;
8) **While** the stack T2 is not empty **Do**
9) { *l* ←Pop an instance from the stack T2;
10) **For** each instance *f* in NT_l **Do**
11) {Create the node *f* which is the child of the node *l*;
12) Delete the instance *f* from the stack T1 and T2;
13) Link the node *f* to the same instance-name non-leaf-node in CPI-tree; }
14) *Push* all the instances in NT_l into the stack T2 in descending order;
}
}
15) Return the root "null"

The procedure *gen_neighborhood* of this algorithm generates a set $NT=\{NT_{l_1}, NT_{l_2}...NT_{l_m}\}$ of spatial neighbor relationships, where NT_{l_i} is the set of neighbor instances of the instance l_i, whose order is "bigger" than l_i and is sorted in ascending order of instances.

In the loop of the step 4), an iteration will create a branch of the root "null" in the CPI-tree, and the loop of the step 8) generates iteratively all nodes of this branch of the CPI-tree. As it happens, the instances are considered in ascending order in the step 10), because the instances are sorted in ascending order in NT_l. The operation deletion in the step 12) is to avoid duplication of information, and the operation linking in the step 13) is for no loss information.

2.2. Complexity and completeness of CPI-tree

Analysis. The computational complexity of the algorithm includes procedure Gen_neighborhood and the rest of algorithm. Suppose the number of spatial instances is *m*. In the worse case, The computational complexity of the procedure Gen_neighborhood will be $O(m^2 \log_2 m)$, and the rest of the algorithm will be $O(m^2)$. We will show that the CPI-tree contains the complete and no redundant information for table instances generating in the following lemmas.

Lemma 1. All the neighbor relationships of given spatial instances are recorded in a CPI-tree, no one is omitted.

Proof: according to the procedure of constructing a CPI-tree, all the spatial instances are scanned and their neighbor relationships are recorded in CPI-tree.

Therefore, none of the spatial instances neighbor relationship is omitted in CPI-tree.

Lemma 2. CPI-tree materializes the neighbor relationships of a spatial dataset with no duplication of the neighbor relationships.

Proof: It is obvious because the step 12) in constructing a CPI-tree algorithm guarantees each spatial neighbor relationship is considered once, and the step 10) and 14) ensure that a connective sub-graph in spatial dataset forms a branch of a CPI-tree.

3. Generating complete table instances using CPI-tree

3.1. Principles of generating table instances from a CPI-tree

Definition 2. (*direct-child-link*). A link between two nodes in a CPI-tree is called as a *direct-child-link*. A k-length direct-child-link is called as a *size-k direct-child-link*. The nodes lying in middle of a size-k direct-child-link are called as *intra-node*. For a node in a direct-child-link, the nodes lying in this node blow are called as *child-node* on the direct-child-link.

Definition 3. (*indirect-child-link*). The child nodes linked out by a dashed link are called as *indirect-child*. A size-k child-link linked out by an indirect-child is called as a size-k *indirect-child-link*.

Definition 4. (*all-link*). If all child-nodes of each intra-node in a size-k direct-child-link are the brothers of corresponding intra-node, then the size-k direct-child-link is called as a size-k *all-link*. If the size-k direct-child-link defined above is a size-k indirect-child-link, then it is called as a size-k *indirect-all-link*.

Based on the definitions above, there are the following properties.

Property 1. (*Child-link property*). Each size-2 child-link in a CPI-tree denotes a size-2 co-location table instance.

This property can be obtained directly from the CPI-tree constructing process, and it is the base of generating other table instances.

Property 2. (*All-link property*). Each size-k all-link or size-k indirect-all-link in a CPI-tree denotes a size-k co-location table instance. (k>2)

Rationale. It is obvious that the instances satisfying all-link or indirect-all-link form a clique in corresponding graph G.

We then show that the complete set of the table instances in a spatial dataset can be generated by using corresponding CPI-tree.

Lemma 3. Table instances generated from the CPI-tree by using **property 1** and **property 2** are correct and complete.

Rationale. First, it is shown that each table instance generated from property 1 and 2 is correct and distinct. The correctness is guaranteed by **property 1** and **property 2**, and the distinction is guaranteed by **lemma 2**.

Second, it is shown that no table instance can be generated out of the CPI-tree. Suppose a size-k table instance can be generated out of the CPI-tree. If this is a size-2 table instance, and then there is not a child-link between the two instances. According to **lemma 1**, there is not a spatial neighbor relationship between the two instances, this reduces to absurdity. For size-k table instances (k>2), that means that these instances presenting to the size-k table instances do not form a all-link or indirect-all-link, and then according to the process of constructing a CPI-tree, there at least is not a neighbor relationship between two instances among these instances. This also reduces to absurdity.

3.2. Table instances generation algorithm

Complete co-location table instances of a spatial dataset are generated recursively from corresponding CPI-tree in the following steps.

1) Initiate prefix pattern α= {null} (the root of CPI-tree) and the suffix pattern β= {null}, search CPI-tree recursively for complete the table instances.

2) If α has a child node *s* and *s* has a child node *t* too, then α is put by *s*, i.e. $\alpha=\{s\}$, search CPI-tree recursively with the new α and β; else (including the last recursion returns), (1). β is put by *s*, i.e. $\beta=\{s\}$, and a table instance $\alpha\cup\beta$ is generated; (2). if β has child node *t*, then (a). Generating a size-3 table instances if the $\alpha\cup\beta\cup\{t\}$ is a all-link, examining the size-4 table instances if *t* has child node…, … (b). If β has indirect-child nodes, examining indirect-all-link to generate some new table instances. For example, consider the node "B.3" in Fig. 2, when β is "B.3", besides generating the table instances "A.3, B.3, C.1" and "A.3, B.3, D.1", the table instance "A.3, B.3, C.1, D.1" also be generated.

3) If the recursion is ended, return all the co-location table instances.

The above approach can be transformed into the following algorithm.

Algorithm 2 (Gen_instance).

Input: *CPI-tree*: materialized spatial neighbor relationships in a spatial dataset and constructed according to Algorithm 1;

Output: The complete set of co-location table instances.

Method: call Gen_instance (CPI-tree, null, null)

Procedure Gen_instance (*CPI-Tree*, α, β)

1) **while** α has child node *s* **Do**

2) {*if s* has child node *t* **then** Gen_instances (CPI-Tree, {s}, β);
3) β= {s};
4) **If** α <> *"null"* **then**
5) { $\alpha \cup \beta$ forms an table instance;
6) *If* β has a child-node or a indirect-child-node **then** *Gen_next_instance (CPI-tree, α, β);* }
7) }

Procedure *Gen_next_instance (CPI-tree, α, β)*
1) { ω= The last element of β;
2) **while** (ω has a child-node / a indirect-child-node *t* & *Ok* (t, β) **Do**
3) { β=$\beta \cup \{t\}$;
4) $\alpha \cup \beta$ forms an table instance;
5) *If t* is not a leaf **then** *Gen_next_instance (CPI-tree, α, β)* }
6) }.

Bool Function *Ok* (t, β) //If the node t is the brothers of β, then return true, else return false
1) { **For** i=1 to $|\beta|$
2) **If** the node *t* is not the brother of the i-th element in β **then** Return false;
3) Return true }

Analysis. With the properties and lemmas in Section 2 and 3, the algorithm correctly finds the complete set of co-location table instances in a spatial dataset.

From the algorithm and reasoning, one can see that the table instances generation process is a backtracking process. The algorithm scans the CPI-tree once and table instances generating is recursively performed on the child-link from size 2. If a size-k table instance is not generated, then the size-k+1 table instance derived from it will not be considered. Moreover, after generating lower table instances, the table instances derived from them will be examined at once. This is much less costly than traditional methods of the full-join and the join-less. Thus the algorithm is efficient. The real execution results will be shown in Section 4.

4. Experimental Results

In this section, the performance of the algorithms is evaluated with the join-less approach using both synthetic and real data sets. All the experiments were performed on a 3-GHz Pentium PC machine with 2G megabytes main memory, running on Microsoft Windows/XP. All programs are written in Java.

The experimental results are reported on two synthetic data sets. The first one is called as sparse dataset with 26 spatial features. In this dataset, when the neighbor distance threshold *d* and the prevalence threshold *min_prev* are set to 25 and 0.15, the total number of size 2 co-locations and the maximum size of co-locations are 104 and 4, respectively, while the number of all instances in the dataset is set to 10k. The prevalent co-locations are short and not numerous in this dataset.

The second synthetic dataset used in the experiments is a dense dataset with 26 spatial features. The total number of size 2 co-locations and the maximum size of co-locations are 232 and 8, when the threshold *d* and the *min_prev* are set to 25 and 0.15, respectively. There exist long prevalent co-locations as well as a large number of short prevalent co-locations in this dataset when the prevalence threshold *Min_prev* goes down.

To test the practicability of CPI-tree, a real dataset, the plant distributing data set of the "Three Parallel Rivers of Yunnan Protected Areas" area, is used. It contains the number of plant species (feature types) is 16. The total number of plant instances is 3908. When *Min_prev* and distance *d* are set to 0.1 and 1900 respectively, the maximum size of co-location is 4 and the total number of size 2 co-location patterns is 42. The characteristic of the dataset is that there are a large number of table instances in each co-location pattern.

1) Scalability with the neighbor distance threshold *d* over sparse data set and dense data set: The runtime of *CPI-tree* and *Join-less* on the sparse and the dense synthetic datasets, when the prevalence threshold *min_prev* is set as 0.5, as the neighbor distance threshold *d* increases from 15 to 25/30 is shown in Fig. 3 and Fig. 4. Since the dataset is sparse, as the threshold *d* is low, the prevalent co-location patterns are short and the set of such patterns is not large, the advantages of CPI-tree over Join-less are not so impressive. However, as the threshold *d* goes up or the dataset becomes dense, the gap becomes wider.

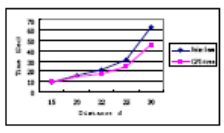


Fig. 3 Scalability with *distance d* over sparse data set

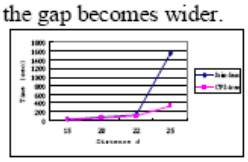


Fig. 4 Scalability with *distance d* over dense data set

2) Scalability with prevalence threshold *Min_prev* over sparse data set and dense data set: Fig. 5 shows the experimental results of scalability with *Min_prev* over the sparse dataset, while the results over dense dataset are shown in Fig. 6. The neighbor distance threshold *d* is set as 200 in the experiments of Fig.5, while d is 150 in the experiments of Fig.6. The advantage of CPI-tree approach is more impressive with threshold *Min_prev* decrease and the dataset becomes dense.

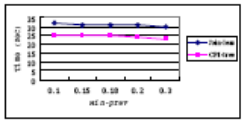


Fig. 5 Scalability with *Min_prev* over sparse data set

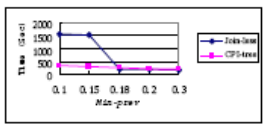


Fig. 6 Scalability with *Min_prev* over dense data set

3) Scalability with prevalence threshold *Distance d* over a real data set: The mining result over a real dataset, a plant distributing data set of the "Three Parallel Rivers of Yunnan Protected Areas" area, is shown in Fig. 7. From the figure, one can see that CPI-tree method is scalable even when there are many table instances. In such real datasets, the join-less method is not comparable to the performance of CPI-tree method.

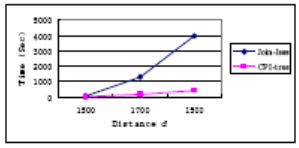


Fig. 7 Scalability with *Distance d* over a plant distributing data set of the "Three Parallel Rivers of Yunnan Protected Areas" area

4) Scalability of CPI-tree algorithm with number of instances: To test the scalability of CPI-tree against the number of instances, the dense dataset is used with *Min-Prev* is set to *0.3,* the neighbor distance threshold *d* is 20, and the number of instances ranges from 3K to 15K. The result is shown in Fig. 8, which shows that the CPI-tree method is the linear increase of runtime with the number of instances.

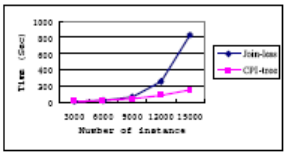


Fig. 8 Scalability of CPI-tree algorithm with number of instances

5. Conclusion and Future Work

In this paper, a new join-less co-location mining algorithm, which can rapidly generate spatial co-location table instances based on the *CPI-Tree* construction materialized neighborhood relationship between spatial instances, was proposed. The algorithm is efficient since it does not require expensive spatial joins or instance join for identifying co-location table instances. The experimental results show the new method outperforms the join-less method in the case of sparse and dense datasets. As future work, the applications studying of co-location patterns mining is an important work. And treat with the redundant co-location rules and maximal co-location patterns mining will be significant works in the future work as well.

6. References

[1] J. Han and M. Kamber, Data mining: concepts and techniques (Second Edition). China Machine Press, Beijing, 2006.

[2] R. Agarwal and R. Srikant, *Fast algorithms for Mining association rules*, In Proc. of Int'l Conference on Very Large Databases (VLDB), 1994.

[3] S. Shekhar, P. Zhang, Y. Huang, and R. Vatsavai, *Trends in Spatial Data Mining, as a book chapter in Data Mining: Next Generation Challenges and Future Directions*, H. Kargupta, A. Joshi, K. Sivakumar and Y. Yesha (editors), AAAI/MIT Press, 2004.

[4] K. Koperski and J. Han, *Discovery of Spatial Association Rules in Geographic Information Databases*, In Proc .of Int'l Symposium on Large Spatial Data bases, Maine, 1995, pp. *47-66*.

[5] L. Wang, K. Xie, T. Chen, and X. Ma, *Efficient discovery of multilevel spatial association rule using partition*, Information and Software Technology (IST), Volume 47, Issue 13, 2005, pp. 829-840.

[6] Y. Huang, S. Shekhar and H. Xiong, *Discovering Colocation Patterns from Spatial Data Sets: A General Approach,* IEEE Transactions on Knowledge and Data Engineering, 2004, 16(12): 1472-1485.

[7] S. Shekhar and Y. Huang, *Co-location Rules Mining: A Summary of Results*, In Proc. of International Symposium on Spatio and Temporal Database (SSTD), 2001.

[8] Y. Morimoto, *Mining Frequent Neighboring Class Sets in Spatial Databases*, In Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2001.

[9] J. S. Yoo and S. Shekhar, *A partial Join Approach for Mining Co-location Patterns*, In Proc. of the 12th annual ACM international workshop on Geographic information systems, 2004, pp. 241-249.

[10] J. S. Yoo, S. Shekhar, and M. Celik, *A Join-Less Approach for Co-Location Pattern Mining: A Summary of Results*, ICDM 2005: 813-816.