FP-TREE BASED SPATIAL CO-LOCATION PATTERN MINING

Ping Yu, B.E.

Thesis Prepared for the Degree of

MASTER OF SCIENCE

UNIVERSITY OF NORTH TEXAS

May 2005

APPROVED:

Yan Huang, Major Professor
Armin R. Mikler, Committee Member
Robert Brazile, Committee Member
and Program Coordinator
Krishna Kavi, Chair of Department of
Computer Science and Engineering
Oscar N. Garcia, Dean of the College
of Engineering
Sandra L. Terrell, Dean of the Robert B. Toulouse
School of Graduate Studies

Yu, Ping, FP-tree Based Spatial Co-location Pattern Mining. Master of Science (Computer Science), May 2005, 68 pp., 13 figures, 8 tables, references, 47 titles.

A co-location pattern is a set of spatial features frequently located together in space. A frequent pattern is a set of items that frequently appears in a transaction database. Since its introduction, the paradigm of frequent pattern mining has undergone a shift from candidate generation-and-test based approaches to projection based approaches. Co-location patterns resemble frequent patterns in many aspects. However, the lack of transaction concept, which is crucial in frequent pattern mining, makes the similar shift of paradigm in co-location pattern mining very difficult. This thesis investigates a projection based co-location pattern mining paradigm. In particular, a FP-tree based co-location mining framework and an algorithm called FP-CM, for FP-tree based co-location miner, are proposed. It is proved that FP-CM is complete, correct, and only requires a small constant number of database scans. The experimental results show that FP-CM outperforms candidate generation-and-test based co-location miner by an order of magnitude.

ACKNOWLEDGEMENTS

This thesis would not have been finished without my research advisor, Dr. Yan Huang. I would like to thank her for her continuous encouragement and support during my work on the thesis. I would also like to thank Dr. Mikler and Dr. Brazile for their help and being on my thesis committee.

I also thank my family and friends for their support throughout these years.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Today's world is the information world. Huge amounts of data exist at every corner of the digital world: World Wide Webs, enterprise databases, data warehouse, etc. At the same time, different types of huge data are coming with an even faster speed. This is really an era of data explosion!

However, data themselves are neither information, nor knowledge. We need to extract, or mine knowledge from large amounts of data. This motivates data mining or knowledge discovery [23, 29, 30, 31], which is the task of discovering interesting patterns from large amounts of data where the data can be stored in databases, data warehouses, or other information repositories. Data mining techniques can be performed on many different kinds of data such as transaction databases, relational databases, data warehouses, spatial databases, multimedia databases, World Wide Web, and etc. The knowledge and information mined can be useful for many applications such as market analysis, business management, engineering design, science exploration, and product control [23]. This thesis mainly discusses a special problem in data mining area: Spatial co-location pattern mining algorithms in spatial databases.

In this chapter, section 1.1 and 1.2 review frequent pattern mining in transaction database and co-location pattern mining in spatial database respectively, comparison of frequent pattern mining and co-location pattern mining is presented in section 1.3, and the contribution and the scope of this thesis are described in section 1.4 and section 1.5 respectively.

1.1 Frequent Pattern Mining

A transaction includes a unique transaction identity number, and a set of items, for example, products purchased together by a customer in a store. A transaction database is a file which consists of records, and each record in the file represents a transaction. The frequent pattern is defined as follows. Let $I = \{i_1, i_2, ..., i_m\}$ be a set of m items, and $D = \{T_1, T_{2,}, ..., T_n\}$ be a transaction database where T_i($\forall$ i$\in$ [1, n]) is a transaction which contains a set of items in *I*. Support of *X* ($X \subset I$) is the probability of a transaction in *D* that contains *X*. *X* is a frequent pattern if *X*'s support is no less than a pre-determined minimum support threshold min_sup. For example, {bread, egg} is a frequent pattern in a transaction database if its support satisfies min_sup.

Mining frequent patterns from large databases is crucial for mining association rules [1] in market basket data, correlations [24], max-patterns [25], sequential patterns [26] and many other interesting data mining tasks. Frequent pattern mining from large databases is an important data mining task. The discovery of frequent patterns and association rules from a transaction database is useful in business decisions [27]. For example, in the management of a

supermarket, the business decisions may include how to design coupons, what to put on sale, and how to place merchandise on shelves to maximize the profit. Therefore, since its introduction, the problem of mining frequent patterns from large databases has been in the subject of numerous studies.

There have been many algorithms developed for efficient and scalable mining of frequent patterns and association rules. In recent years, the paradigm of frequent pattern mining algorithms has undergone a fundamental shift from candidate generation-and-test based approaches to projection based approaches. Apriori algorithm [1] finds frequent patterns using candidate generation-and-test approach. It employs an iterative approach known as level-wise search, where frequent k-patterns are used to explore frequent (k+1)-patterns. FP-growth [6] is a projection based approach of mining frequent patterns without candidate generation-and-test. It compresses the original transaction database into a highly compact data structure (an FP-tree). Rather than employing the candidate generation-and-test strategy of apriori algorithm, it focuses on frequent pattern growth which avoids costly database scans and candidate generation, resulting in greater efficiency.

1.2 Co-location Pattern Mining

Spatial data [9] is the data related to objects that occupy space. Spatial databases are special databases which contain spatial-related information [37]. They include geographic databases, satellite image databases, and other databases related to the storage of spatial data [32, 36]. Spatial data mining is

the extraction of implicit knowledge, spatial relations, or patterns not explicitly stored in spatial databases [10, 34]. In recent years, advanced data collection tools such as global positioning systems (GPS) [33] and earth's observing systems (EOS) are accumulating increasingly large spatial databases [35]. Meantime more reliable, powerful, and inexpensive location-enabled mobile devices are also generating large geo-referenced datasets. Efficient tools for spatial data mining are very important for the organizations which make decisions based on large spatial databases, such as ecology and environmental management, public health, and transportation [39]. This emphasizes the need for an automatic discovery of spatial knowledge. The automatic discovery of interesting, potentially useful, and previously unknown patterns from large spatial datasets has been widely investigating via various spatial data mining [14, 15, 9, 19] techniques. Classical spatial patterns include spatial clustering [13, 45, 44], spatial characterization [3], spatial outlier detection [17], spatial prediction [18], spatial classification [46], and spatial boundary shape matching [8].

This thesis focuses on a recent spatial data mining problem: Finding sets of spatial features that tend to be located in spatial proximity. This problem is also referred to as spatial co-location pattern mining [10, 5, 2, 16, 12, 38, 40, 42, 43], which is an important spatial data mining task with broad applications. Examples include diseases and related health hazards in nearby region in epidemiology, e.g. the contaminated water reservoirs with a certain disease in their spatial neighborhood, predator-prey species and symbiotic species in ecology, e.g. the Egyptian plover and Nile crocodile, and high rate of car accidents and associated

road/sign configuration in transportation, e.g. short highway exit ramp, service road, and “yield” sign implies a high rate of car accidents in its nearby region. For example, in figure 1.1, the locations of nine spatial features are given. Feature 1 represents a certain disease, feature 3 represents a chemical factory. Is {1, 3} a co-location pattern?

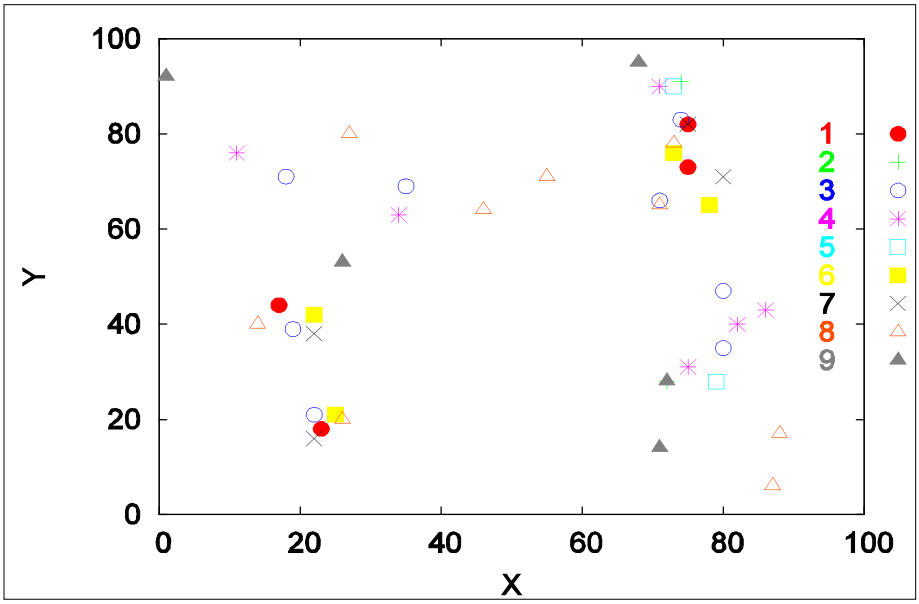


Figure 1.1 A Spatial Dataset

Many algorithms for co-location mining proposed in literature [10, 5, 16, 12, 4] employ a candidate generation-and-test co-location mining paradigm, which utilizes the anti-monotone property of interestingness measures. In a clustering-based map overlay approach [5, 4], every spatial feature is treated as a map layer and point-data in each layer are clustered into regions. Given X as a set of map layers, the clustered support of X is defined as the ratio of the overlapping area of regions that satisfies X. Clustered support is anti-monotone in nature and a level wise algorithm is applied after clusters in each layer are approximated by

polygons. In a reference feature based approach [10], transactions are created around each instance of a user-specified spatial feature using neighbor relationship R. Each transaction contains all the spatial features in the neighbor relationship R of the current instance. After transactionization, a level wise algorithm is applied. Under this model, a frequent pattern based algorithm can be applied straightforwardly due to the fact that the interestingness measure is defined based on the generated transactions. In the distance based approaches [12, 16], the instances of each spatial feature set define the interestingness measure. An instance of a spatial feature set X is defined as a set of instances of X that form a clique under a given neighbor relationship R. The number of instances for each pattern does not possess an anti-monotone property by nature. The k-neighboring set model [12] uses a non-overlapping-instance constraint to get the anti-monotone property for this measure. Frequent pattern mining is not required once disjoint instances are obtained. In an event centric model [16], a participation index of a pattern, which is the minimal participation ratio of the instances of each feature in the pattern, is defined as the interestingness measure. FC [20] uses hash-based spatial join techniques [7] and multi-way spatial joins [11] to mine spatial co-locations.

1.3 Comparison of Frequent Pattern Mining and Co-location Pattern Mining

Spatial co-location patterns resemble frequent patterns in many aspects. However, there are some differences between them. The comparison of frequent pattern (FP) mining and co-location pattern mining is shown in table 1.1.

Frequent Pattern Mining	Co-location Pattern Mining
Item	Spatial feature
Item set	Spatial feature set
Frequent pattern	Co-location pattern
Support	Spatial interestingness measure
Transaction database	Spatial database

Table 1.1 Comparisons of FP Mining and Co-location Pattern Mining

The paradigm of frequent pattern mining algorithms has shifted from candidate generation-and-test based approaches to projection based approaches. Projection based approaches have major advantages over candidate generation-and-test based approaches and avoid costly database scans by compressing transaction data into compact structures. A finite set of disjoint transactions is crucial in frequent pattern definition and its mining algorithms. Spatial co-location patterns resemble frequent patterns in many aspects. However, the difficulty in creating explicit disjoint transactions from continuous spatial data makes the similar paradigm shift in spatial co-location pattern mining very difficult. The lack of pre-materialized transactions becomes a major obstacle in adopting projection based algorithms into spatial co-location pattern mining.

A natural question to ask is: The paradigm shift in frequent pattern mining can be pushed for mining spatial co-location patterns?

1.4 Contribution of this Thesis

A FP-tree based co-location mining framework is proposed in this thesis. This framework can incorporate any fast maximal frequent pattern mining

algorithm. This framework is designed as an open, scalable framework for the research of spatial co-location pattern mining, as well as software development. A fully standardized framework can integrate any algorithm to be evaluated and investigated.

A FP-tree based co-location miner called FP-CM algorithm is also proposed based on the proposed framework. It combines the salient features of FP-tree based maximal frequent pattern mining [6, 21] and fast multi-way spatial joins [11] to reduce the total number of database scans to five.

Finally, FP-CM algorithm's completeness, correctness, and efficiency are proved. The experimental results also show that FP-CM is an order of magnitude faster than a candidate generation-and-test based spatial co-location miner CM.

1.5 Scope of this Thesis

This thesis reviews some basic concepts and background in frequent pattern mining and spatial co-location pattern mining, and focuses on the spatial co-location pattern mining problem. A novel framework and an efficient algorithm are designed for spatial co-location pattern mining and a series of experiments are conducted to test it in comparison with one previous work.

The organization of the rest of the thesis is as follows. Chapter 2 recalls some important concepts and investigates a candidate generation-and-test based frequent pattern mining, a projection based frequent pattern mining paradigm in frequent pattern mining. Chapter 3 reviews the basic concepts and a candidate generation-and-test based spatial co-location pattern mining paradigm,

and a multi-way spatial joins based co-location pattern mining paradigm in spatial co-location pattern mining. Chapter 4 proposes a FP-tree based co-location mining framework and an algorithm called FP-CM, for FP-tree based co-location miner. Chapter 5 analyzes its completeness, correctness, computational efficiency, and presents the experimental evaluation. This thesis is summarized in chapter 6.

CHAPTER 2

FREQUENT PATTERN MINING

A frequent pattern [23] is a pattern (such as a set of items, subsequence, or substructure) that frequently occurs in a database. A set of items such as bread and eggs that are often purchased together is a frequent pattern. Mining frequent patterns from large databases is crucial for mining association rules [1] in market basket data. Association rule mining [47] is an important data mining technique, which helps managers find frequent patterns to design store layouts, plan catalogs, and decide which items to put on sale. For example, if customers often purchase digital cameras and printers together, then having a sale on digital cameras may increase the sale of printers as well as digital cameras. Therefore, frequent pattern mining is an important data mining task in data mining and mining frequent patterns from large databases has been studied by many researchers in data mining area.

This chapter introduces the basic concepts and two classic algorithms in frequent pattern mining. The basic concepts of frequent pattern mining are given in section 2.1. Section 2.2 presents the apriori algorithm, a candidate generation-and-test algorithm for mining frequent patterns. An efficient frequent pattern mining algorithm without candidate generation, called FP-growth is presented in

section 2.3. Section 2.4 describes the performance comparison of FP-growth and apriori algorithm. This chapter is summarized in section 2.5.

2.1 Basic Concepts

2.1.1 Terminology in Frequent Pattern Mining

Let $I = \{i_1, i_2, ..., i_m\}$ be a set of m items, and $D = \{T_1, T_{2,}, ..., T_n\}$ be a set of n transactions. Each transaction in D is a set of items such that $T_i \subseteq I \ \forall$ i $\in$ [1, n]. We say a transaction T_i contains X if and only if $T_i \supseteq X$. Here X is a set of items in I . Some basic terminologies related to frequent pattern mining are introduced.

- An association rule is an implication $X \rightarrow Y$, where $X \subset I$, $Y \subset I$, and $X \cap Y = \phi$.
- Support s of $X \rightarrow Y$ is the probability of a transaction in D that contains both X and Y . That is $Support(X \rightarrow Y) = P(X \cup Y)$.
- Confidence c of $X \rightarrow Y$, is the conditional probability that a transaction in D containing X that also contains Y .That is $confidence(X \rightarrow Y) = P(Y \mid X)$.
- The min_sup is a pre-determined minimum support threshold. The min_conf is pre-determined minimum confidence threshold. Rules with $P(X \cup Y) \geq$ min_sup and $P(Y \mid X) \geq$ min_conf are called strong.
- A pattern is a set of items. A pattern that contains k items is called a k-pattern. If support of a pattern is greater than or equal to min_sup**,** then it is a frequent pattern. Support count of a pattern is the number of transactions that contain the pattern in D .

2.1.2 Steps of Association Rule Mining

From the definition of confidence, we can generate the formula as:

$$\begin{aligned} &Confidence(X \rightarrow Y) \\ &= P(Y \mid X) \\ &= P(X \cup Y) / P(X) \\ &= \sup port\_count(X \cup Y) / \sup port\_count(X). \end{aligned}$$

The formula shows that we can obtain $Confidence(X \rightarrow Y)$ by knowing the support or support counts of $X \cup Y$ and X. That means, if the supports or support counts of $X \cup Y, Y$ and X are found, their association rules can be easily derived. Therefore, mining association rules can be reduced to mining frequent patterns.

Therefore, association rules can be mined in two steps. The first step is mining frequent patterns. The second step is generating strong association rules from the frequent patterns. The first step is much more costly than the second one in terms of computation, so the overall performance of association rule mining is determined by the first step.

For example, a transaction database is in table 2.1. Let min_sup = 50%, min_conf = 50%. From table 2.1, the frequent patterns and their supports can be obtained in table 2.2.

Transaction	Items purchased
T_1	b, d, g
T_2	a, c, d
T_3	a, c
T_4	d, e, f

Table 2.1 A Transaction Database

Frequent Pattern	Support
{a}	50%
{c}	50%
{d}	75%
{a, c}	50%

Table 2.2 Frequent Patterns and their Supports

For association rule $\{a\} \rightarrow \{c\}$, we can get

$$Support(\{a\} \rightarrow \{c\}) = P(\{a\} \cup \{c\}) = 50\%,$$

$$confidence(\{a\} \rightarrow \{c\}) = P(\{a\} \cup \{c\}) / P(\{a\}) = 50\% / 50\% = 100\%.$$

2.1.3 Maximal Frequent Pattern and Closed Frequent Pattern

Mining complete frequent patterns in a large database is a significant challenge. Since all non-empty subsets of a frequent pattern are frequent patterns, mining complete frequent patterns in a large database often generate a huge number of patterns that satisfying the min_sup, especially when the min_sup is set low. A long pattern contains a combinatorial number of shorter frequent patterns. For example, a frequent 100-pattern such as $P_1 P_2 ... P_{100}$, contains $\binom{1}{100} + \binom{2}{100} + ... + \binom{100}{100} = 2^{100} - 1 \approx 1.27 \times 10^{30}$ frequent patterns, which are too huge to be enumerated for a computer. In order to solve this problem, maximal frequent pattern (MFP) is introduced.

A pattern X is called a MFP in the database if X is frequent, and there exists no superset Y such that $X \subset Y$ and Y is a frequent pattern in the database.

For example, given a transaction database $D = \{\{i_1, i_2, ..., i_{50}\}, \{i_1, i_2, ..., i_{100}\}\}$, let min_sup = 50%. What are MFPs in *D*?

There is only one MFP $\{i_1, i_2, ..., i_{100}\}$ in the database *D* and its support is 50%, whereas there are $\binom{1}{100} + \binom{2}{100} + ... + \binom{100}{100} = 2^{100} - 1 \approx 1.27 \times 10^{30}$ complete frequent patterns, which are too many frequent patterns for a computer to compute and store.

However, the problem is that MFPs do not contain the complete information of all the frequent patterns. In the above example, from the MFP and its support, we can only assert that all its subsets are frequent patterns and any of their supports is greater than or equal to 50%, but we do not know their exact supports. In order to solve this problem, closed frequent pattern (CFP) is introduced.

A pattern X is a CFP in the database if X is frequent, and there exists no superset Y such that $X \subset Y$ and Y has the same support as X in the database. The CFPs contain the complete information of all the frequent patterns. In general, although the number of CFPs is greater than the number of MFPs, it is still much smaller than the number of the complete frequent patterns.

For example, given a transaction database $D = \{\{i_1, i_2, ..., i_{50}\}, \{i_1, i_2, ..., i_{100}\}\}$, let min_sup = 50%. What are CFPs in *D*?

We can get two CFPs, which are $\{i_1, i_2, ..., i_{50}\}$(support is 100%) and $\{i_1, i_2, ..., i_{100}\}$ (support is 50%). The two CFPs contain the complete information of all the frequent patterns. From them, we can derive all the frequent patterns in *D*.

For example, we can derive $\{i_5, i_{38}\}$ (support is 100%) since $\{i_5, i_{38}\}$ is a subset of frequent pattern $\{i_1, i_2, ..., i_{50}\}$ (support is 100%) and $\{i_{10}, i_{68}\}$ (support is 50%) since it is a subset of frequent pattern $\{i_1, i_2, ..., i_{100}\}$ (support is 50%).

2.2 Apriori Algorithm

Apriori [1] is an efficient association rule mining algorithm, which finds frequent patterns using candidate generation-and-test approach.

Two important concepts in apriori algorithm:

- Apriori property: All nonempty subsets of a frequent pattern must be frequent.

  For example, if {bread, milk} is frequent, so is {milk}. That is because every transaction containing {bread, milk} must also contain {milk}.
- Apriori pruning principle: If there is any pattern which is not frequent, then all of its supersets should not be frequent and should not be generated and tested. For example, if a pattern X is not frequent, then $P(X)$ < min_sup. If a pattern Y is added to the pattern X, because $P(X \cup Y) < P(X)$, $P(X \cup Y)$ < min_sup. Therefore, $X \cup Y$ is not frequent either.

The ideas of apriori algorithm:

Apriori algorithm [1], a candidate generation-and-test based approach, is an efficient association rule mining algorithm that explores the level-wise mining Apriori property used to improve the efficiency. At the kth iteration, it forms

frequent (k+1)-pattern candidates based on the frequent k-patterns, and scan the database once to find the frequent (k+1)-patterns. It works as follow. First, the set of frequent 1-patterns L_1 is found. Next, L_1 is used to find the set of frequent 2-patterns L_2, which is used to find L_3, and so on, until L_n is empty. Since finding each L_i requires one database scan, the total number of database scan is n for finding L_n.

The bottlenecks of apriori algorithm:

- Mining long patterns needs many database scans and checks a huge number of candidates by pattern matching. For example, in order to find frequent 100-pattern $P_1P_2\ldots P_{100,}$ it needs 100 database scans.
- It may need to generate a large number of candidates, compute their supports, and test. For example, in order to find frequent 100-pattern $P_1P_2\ldots P_{100,}$ in the worst case, it needs to generate $1.27*10^{30}$ candidates, compute their supports, and test.

Can we reduce the number of database scans and avoid the candidate generation? FP-growth [6] presented in the next section can solve these problems.

2.3 FP-growth Algorithm

Frequent-Pattern growth, called FP-growth [6], is an efficient FP-tree (projection) based frequent pattern mining algorithm without candidate generation.

The ideas of FP-growth algorithm:

FP-growth [6], a projection based approach, is a method of mining frequent patterns without candidate generation-and-test. It employs a divide-and-conquer strategy as follows: Compress the original transaction database into a highly compact data structure (an FP-tree), but retain the pattern association information, and then divide the compressed database into a set of conditional databases (a special kind of projected database), where each conditional database is associated with one frequent 1-pattern, and mine each conditional database separately. Rather than employing the candidate generation-and-test strategy of apriori algorithm, it focuses on frequent pattern growth which avoids costly database scans and candidate generation, resulting in greater efficiency. The FP-growth only needs two database scans when mining all frequent patterns. The first database scan counts the supports of all the 1-patterns, derives the set of frequent 1-patterns, which is sorted in the descending order of their supports. The second database scan creates the initial FP-tree, which contains all support information of the original database.

How to create the FP-tree?

- First, create the “null” root of the FP-tree. The ordered frequent 1-patterns are inserted in the header table.
- The set of frequent 1-patterns in each transaction is inserted into the FP-tree as a branch in the descending order of their supports. If a pattern shares a prefix of a branch in the FP-tree, the new pattern will share the prefix of the branch.

How does FP-growth get the efficiency?

- A large database is compressed into a highly compact FP-tree, which saves costly database scans in the subsequent mining processes.
- FP-tree based frequent mining employs a pattern growth method to avoid the costly generation and test of a huge number of candidates.
- A partitioning-based, divide-and-conquer method is used to reduce the size of subsequent conditional pattern bases and conditional FP-trees.

2.4 Performance Comparison of FP-growth and Apriori

Figure 2.1 shows the scalability of FP-growth and apriori as the support threshold decreases from 3% to 0.1% [6]. FP-growth is about an order of magnitude faster than apriori as the support threshold goes down in large database. This is because the number and length of frequent patterns increase dramatically as the support threshold reduces. The number of candidates that apriori needs to generate and test becomes extremely huge. Therefore, the pattern matching with the large number of candidates by searching the large transaction database becomes very expensive when the support threshold goes down.

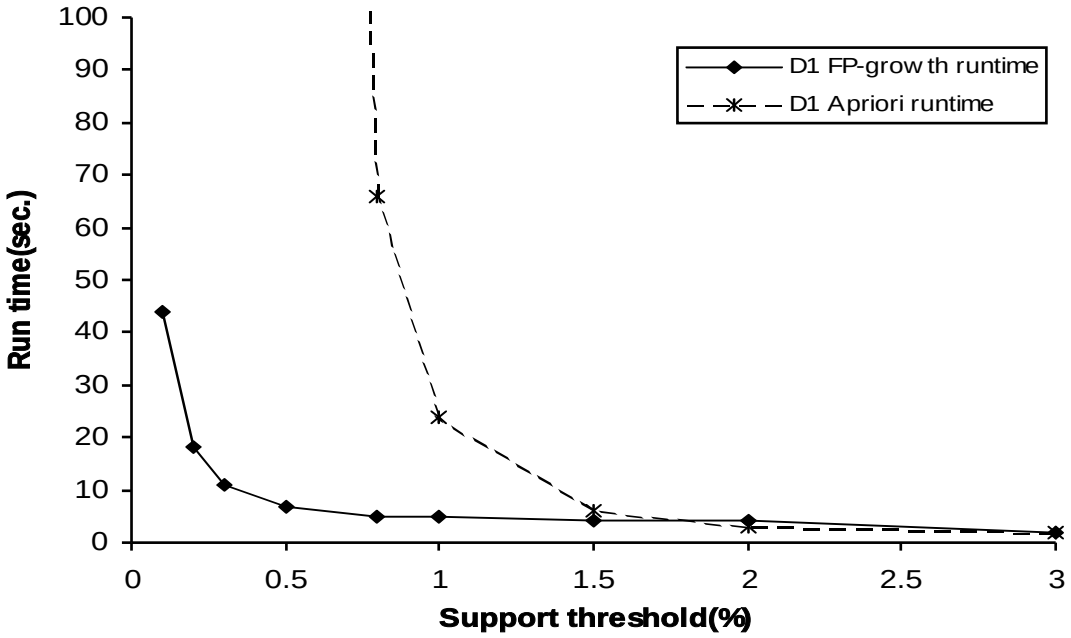


Figure 2.1 Performance Comparison of FP-growth and Apriori

2.5 Summary

Frequent pattern mining is an important data mining task in data mining. To mine association rule we first need to mine frequent patterns, from which strong association rules in the form of $X \rightarrow Y$ can be generated for later knowledge extraction. In this chapter the basic concepts in frequent pattern mining are reviewed. There have been many algorithms developed for efficient and scalable mining of frequent patterns and association rules. Two classic frequent pattern mining methodologies, which are based on candidate generation-and-test and projection-based respectively, are reviewed.

Apriori algorithm is an efficient association rule mining algorithm that explores the level-wise mining apriori property: All nonempty subsets of a

frequent pattern must also be frequent. At the kth iteration (for $k \geq 1$), it forms frequent (k+1)-pattern candidates based on the frequent k-patterns, and scan the database once to find the complete set of frequent (k+1)-patterns, L_{k+1}.

FP-growth is a method of mining frequent patterns without candidate generation. It constructs a highly compact data structure (an FP-tree) to compress the original transaction database. Rather than employing the candidate generation-and-test strategy of apriori algorithm, it focuses on frequent pattern growth which avoids costly candidate generation, resulting in greater efficiency.

CHAPTER 3

SPATIAL CO-LOCATION PATTERN MINING

A spatial co-location pattern is a set of spatial features which are frequently located together in spatial proximity. Spatial co-location patterns resemble frequent patterns in many aspects. Spatial association rules [10] are special cases of general association rules where at least one of the predicates is spatial. In general association rule mining, a finite set of disjoint transactions is input into the association rule mining algorithms [1, 27] such as apriori [1] and FP-growth [6] to find frequent patterns, and association rules from these frequent patterns found. However, spatial association rules mining is different from association rules mining because of the lack of natural transactions due to the continuous spatial data. This creates difficulty in mining spatial association rules.

There are three different models for mining spatial association rules, which are reference feature centric model [10], window centric model, and event centric model [16]. Transactions can be generated using reference feature centric model [10] or window centric model. However, these methods may yield underestimate or overestimate of support or confidence which is used in association rule mining algorithms. Therefore, co-location pattern mining is crucial for mining spatial association rules. It has been studying by many

researchers in spatial data mining research and is an essential data mining task in spatial data mining.

This chapter reviews the background of spatial co-location mining. The problem formulation and basic concepts are given in section 3.1. Section 3.2 reviews the reference feature centric model. Section 3.3 describes the co-location miner [16]. A fast co-location miner [20] is presented in section 3.4. This chapter is summarized in section 3.5.

3.1 Problem Formulation and Basic Concepts

3.1.1 Problem Formulation

The spatial co-location mining problem is formalized below.

Given:

- A set of spatial features $F = \{1,2,...,k\}$.
- A set of n instances $I = \{i_1, i_2, \ldots, i_n\}$, each $i_k \in I$ is a vector <instanceID, spatial feature f, location (x,y)>, where $f \in F$ and $(x,y) \in$ spatial framework S.
- A neighbor relationship R over locations in S.
- Prevalence threshold.

Find:

- Co-locations with prevalence no less than the given threshold.

Objectives:

- Completeness: A spatial co-location mining algorithm is complete if it finds all spatial co-locations with prevalence no less than the given threshold.
- Correctness: A spatial co-location mining algorithm is correct if any spatial co-location it finds has prevalence no less than the given threshold.
- Computational efficiency: IO and CPU cost to find the co-locations should be acceptable.

3.1.2 Basic Concepts

Let $F = \{1,2,...,k\}$ be a set of spatial features, $SD = \{SD_1, SD_2, ..., SD_k\}$ be a set of k spatial datasets such that $\forall i \in [1,k], SD_i$ contains all and only the instances of the spatial feature i, and R be a given spatial neighbor relation (e.g. distance no more than 2 miles). For example, in figure 3.1, the set of all spatial features is $F = \{a, b, c\}$. Feature a has 3 instances in the spatial dataset, i.e. $SD_a = \{a_1, a_2, a_3\}$. Similarly, $SD_b = \{b_1, b_2, b_3\}$, and $SD_c = \{c_1, c_2\}$. The spatial neighbor relationship R is a Euclidean distance and neighboring instances are connected in Figure 3.1.

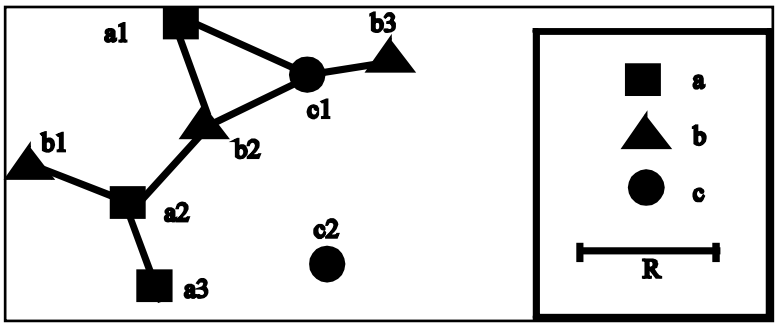


Figure 3.1 A Spatial Database

An m-pattern X is defined a set of spatial features $X \subseteq F, X = \{1,2,...,m\}$.

A set of instances $\{o_1, o_2,..., o_m\}$ is an instance of X if and only if

$$\forall 1 \le i \le m, o_i \in SD_i \text{ ^ } \forall 1 \le i \le j \le m, o_j \in SD_j, dist(o_i, o_j) \le R.$$

It means all pairs of instances in an instance of X should be close to each other. In other words, the relationships between the instances of an instance of X form a clique graph. In figure 3.1, for the pattern $X = \{a,b,c\}$, $\{a_1, b_2, c_1\}$ is an instance of X. However, $\{a_1, b_3, c_1\}$ is not an instance of X because a_1 and b_3 are not close to each other.

The $pr(X, f)$, participation ratio of feature f in a pattern X is defined as

$$pr(X, f) = \frac{\#\ of\ instances\ of\ f\ that\ participate\ in\ any\ instance\ of\ X}{\#\ of\ instances\ of\ f}.$$

The $pi(X)$, participation index of a pattern X is defined as

$$pi(X) = \min\{pr(X, f),\ \forall f \in X\}.$$

For example, in figure 3.1, the instances of spatial feature set $\{a,b\}$ are $\{a_1, b_2\}, \{a_2, b_1\}$ and $\{a_2, b_2\}$. The participation ratio of a in {a, b}, $pr(\{a,b\}, a)$

is $2/3$, since only 2 instances a_1 and a_2 of 3 instances of spatial feature a participate in the instances of $\{a,b\}$. Thus, the participation index of $\{a,b\}$,

$$\begin{aligned} pi(\{a,b\}) &= \min\{pr(\{a,b\},a), pr(\{a,b\},b)\} \\ &= \min\{2/3,\ 2/3\} \\ &= 2/3. \end{aligned}$$

A co-location pattern is a set of spatial features frequently located together in space. If the participation index of a pattern is not less than the user-specified participation index threshold min_pi, then it is a co-location pattern.

The participation index pi is monotonic non-increasing with the size of the co-location increasing. If $X \subseteq X'$, then $pi(X) \geq pi(X')$. The reason is that any feature participates in an instance of X' should also participate in an instance of X. Therefore, apriori property holds for pi of co-locations.

A co-location rule is of the form $C_1 \rightarrow C_2(p, cp)$ where C_1 and C_2 are co-locations, p is a number representing prevalence and cp is a number representing conditional probability. The interest measures (p and cp) may be defined differently in different models.

3.2 Reference Feature Centric Model

The reference feature centric model [10] is relevant to application domains focusing on a specific spatial feature such as city. It converts spatial data to a set of transactions around instances of the reference spatial feature. Given feature i, a transaction database TD_i can be defined as follows:

$k = 1$;
for each instance $o_i \in SD_i$ {
$T_k = \{ j \mid j \neq i \ \wedge \ \exists o_j \in R_j, dist(o_j, o_i) \leq R \}$;
$k++$;
}
$TD_i = \cup_k T_k$; .

A transaction database reference to feature i, TD_i, can be input into a frequent pattern mining algorithm to get features that occur frequently close to feature i.

For example, in figure 3.1, the shapes represent different features, and the lines indicate instances pairs within distance R from each other. For each instance of a, we generate a transaction; a_1 generate $\{b, c\}$ because there is at least one instance of $b(b_2)$ and one instance of $c(c_1)$ close to a_1. Similarly, a_2 generate $\{b\}$ because there is at least one instance of b (e.g., b_1 and b_2) close to a_2. Therefore, we can get $TD_a = \{\{b, c\}, \{b\}\}$. Let min_sup = 30%, we can generate frequent patterns $FP_a = \{\{b\}, \{c\}, \{b, c\}\}$ in TD_a using algorithms such as Apriori [1] and FP-growth [6].

3.3 Co-location Miner

Because of the downward closure property of the participation index [16], a generation-and-test mining paradigm is employed by previous algorithms, e.g. co-location miner, called CM [16], which is event centric approach. CM mines spatial co-location patterns using event centric model which replaces transactions by neighborhoods. The event centric model [16] is relevant to applications such as ecology where there are many boolean spatial features.

Ecologists are interested in mining spatial co-location patterns, which are feature sets frequently to occur in a neighborhood. In an event centric model [16], participation index pi is defined as the interestingness measure.

The ideas of CM algorithm:

- It first uses spatial join [22] to get the instance pairs within distance R to each other and produce instances of candidate size 2 co-locations. Then, the participation index for each candidate size 2 co-location is calculated, and it is accepted as a size 2 co-location if its pi is not less than min_pi. After pruning based on pi, only prevalent co-locations and their instances are kept.
- From two size k co-locations C_1 and C_2 where only the first (k-1) features are common and the last ones are different, a candidate size k+1 co-location C whose subsets are all co-locations is generated based on apriori_gen [1].
- After getting all instances of C by spatial joining on the instances of C_1 and C_2 where only the first k-1 feature instances are common and the distance of instances of the last features is at most R, the participation index is calculated to validate whether C is a co-location.

For example, in figure 3.1, let min_pi be 1/3. After spatial joining, we can get the instances of $(a,b),(a,c)$ and (b,c) as shown in table 3.1. Because each pi satisfies min_pi, they are all co-locations shown in table 3.1. From $\{a,b\}$ and $\{a,c\}$, we can get candidate size 3 co-location $\{a,b,c\}$, whose all size 2

subsets are co-locations, and its instance and pi shown in Table 3.1. Because its pi satisfies min_pi, we can get $\{a, b, c\}$ is a size 3 co-location.

Size	Candidate Co-locations	Instances	Participation Index	Co-location? (min_pi=1/3)
2	{a, b}	$\{a_1,b_2\},\{a_2,b_1\},\{a_2,b_2\}$	min{2/3,2/3}=2/3	Yes
	{a, c}	$\{a_1,c_1\}$	min{1/3,1/2}=1/3	Yes
	{b, c}	$\{b_2,c_1\},\{ b_3,c_1\}$	min{2/3,1/2}=1/2	Yes
3	{a, b, c}	$\{ a_1,b_2,c_1\}$	min{1/3,1/3,1/2}=1/3	Yes

Table 3.1 Co-location Miner Algorithm Illustration

CM has some efficiency problems:

- It requires a potentially large number of instances of size (k-1) co-locations to be computed before size k co-locations can be found. These instances can be too many to be fit in memory.
- The computational cost of processing them and computing participation ratios for them is very high.
- Many spatial joins are needed to find the instances of candidate co-locations and many database scans needed to discovery all the co-locations.

To solve the efficiency problems of CM, a fast co-location miner, called FC was proposed in [20], which integrates the computation of spatial neighbor relationships with the mining algorithm and is presented in next section.

3.4 Fast Co-location Miner

FC [20] uses hash-based spatial join techniques [7] and multi-way spatial joins [11] to mine spatial co-locations, which is much faster than CM when the

total number of features is small (no more than a few dozens). It extends multi-way spatial join algorithms to discover instances of any size co-locations and calculate the participation ratios of features in them, which can be used to compute the participation indexes of them to mine the prevalent co-locations. Spatial joins play an important role in effective spatial query processing. A pair-wise spatial join combined two datasets on a spatial predicate. An example is, “Find all cities that are close to a forest.” Several algorithms were proposed for it. An algorithm proposed in [7] hashes two spatial datasets into buckets using a grid, join bucket pairs to find the join pairs that qualify a spatial predicate. Multi-way spatial joins combine more than two datasets with respect to some spatial predicate. An example is, “Find all cities that are close to forests which are crossed by a river.” Algorithms proposed in [11] extend the pair-wise spatial join algorithms to apply on multiple datasets join.

The ideas of FC algorithm [20]:

- FC algorithm includes two steps. One is hashing step, which imposes a grid over the space and each instance of each feature is hashed into the cell(s) intersected by the disk centered at the instance with radius R. Another is mining step, which employs a multi-way main memory spatial join algorithm based on plane sweep [11, 28] to efficiently find the co-locations.
- FC algorithm needs two database scans: One for hashing step and one for reading the partitions.

The bottleneck of FC:

- FC counts the instances of the power-set of all possible co-locations at one scan, so the space required is too big when the total number of features is large, which makes FC slow.

To solve the bottleneck of FC, an algorithm FP-CM is proposed in the next chapter.

3.5 Summary

Some basic concepts of spatial co-location pattern mining and two algorithms CM and FC for mining co-location patterns are reviewed. CM is a generation-and-test mining, non-transaction algorithm, which mines spatial co-location patterns using event centric approach. This approach generates candidate size k+1 co-location set based on size k co-location set. However, CM has some efficiency problems. FC [20], which solve the efficiency problem of CM, uses hash-based spatial join techniques [7] and multi-way spatial joins [11] to mine spatial co-locations, which is much faster than CM when the total number of features is small (no more than a few dozens). However, FC is slow because the space required is too high when the total number of features is large.

To solve these problems, a new novel framework and an algorithm called FP-CM are proposed in chapter 4.

CHAPTER 4

FP-TREE BASED SPATIAL CO-LOCATION PATTERN MINING

Projection based frequent pattern mining, e.g. FP-growth [6], utilizes a highly condensed FP-tree structure to compress frequent patterns and employs a pattern fragment growth method for mining the complete set of frequent patterns from the FP-tree. Due to the reduced number of database scans and no candidate generation-and-test, this algorithm is very fast in comparison with traditional candidate generation-and-test algorithms [1]. However, a FP-tree based algorithm can not be used directly on spatial co-location mining because of the lack of transactions in spatial datasets. Creating transactions for spatial datasets and establishing the relationship between support and participation index to develop a complete and correct FP-tree based co-location mining algorithm are non-trivial.

In this chapter, our effort to mine the spatial co-location patterns is introduced. A FP-tree based co-location mining framework is proposed in section 4.1. Based on the idea of the framework, a FP-tree based co-location miner, called FP-CM is proposed in section 4.2. This chapter is summarized in section 4.3.

4.1 FP-tree Based Co-location Mining Framework

The proposed framework consists of four function modules as shown in figure 4.1.

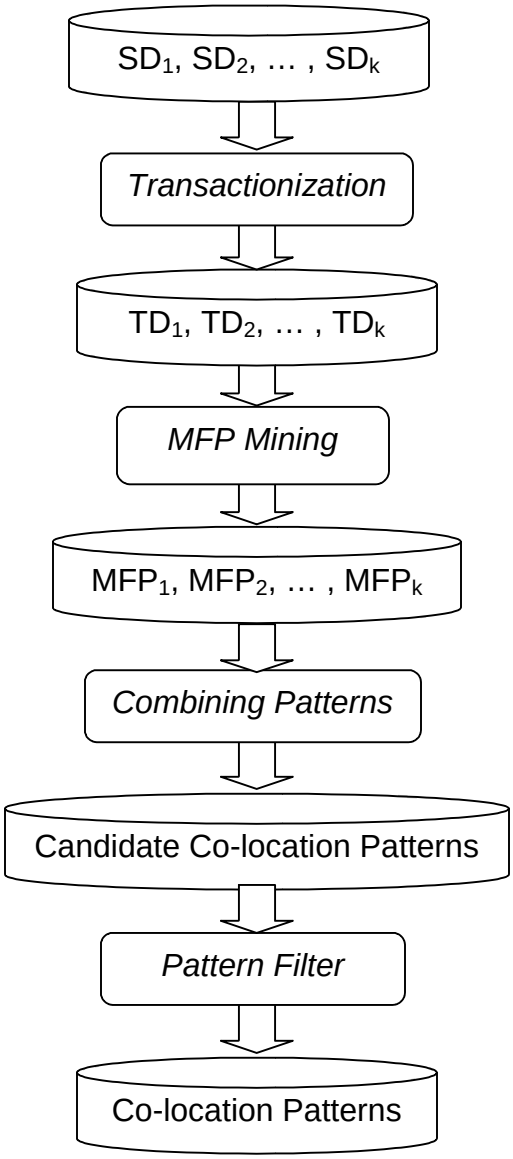


Figure 4.1 FP-tree Based Co-location Mining Framework

- Transactionization: For each available spatial feature in spatial database, a transaction database TD_i is created by the transactionization module.
- Maximal frequent pattern mining: For each transaction database TD_i, we can employ suitable maximal frequent pattern mining algorithm to find maximal frequent patterns MFP_i for later processing.
- Combining patterns: In this module, mined maximal frequent patterns MFP_i are combined together to generate a superset of all the co-location patterns.
- Pattern filter: The results from Combining Pattern could include some false co-locations. A pattern filter is necessary to filter out these false co-locations.

The challenges of implementing the ideas of the proposed framework:

- How to generate transactions? Transactions can be produced by partitioning the space disjointedly or overlappingly. However, disjoint partitioning may lose neighboring instances across boundaries, leading to the loss of potential co-location patterns. Overlapping partitioning may double count instances in two partitions, leading to produce false co-location patterns. The available transaction generation algorithm could be based on window centric model [16] or reference feature centric model [10].

- How to find co-location patterns from the mined maximal frequent patterns generated from the transaction databases? How to assemble the maximal frequent patterns into co-locations?
- How to guarantee the algorithm is complete, correct, and efficient?

The advantages of the proposed framework:

A generic framework could make us easier to get the whole picture of the spatial co-location mining procedure and tasks. Under the framework, the spatial mining problem can be solved independently. For example, any fast maximal frequent pattern mining algorithm [1, 6, 41] can be applied to each transaction database TD_i to find maximal frequent patterns MFP_i without any effect on previous modules or later processing. The interfaces between the four function modules can be standardized, therefore any algorithm follows the standard data format could be easy to be integrated into the framework for investigation and test. This framework can be defined and implemented as an open and scalable framework which could benefit the later research efforts, as well as real software development.

4.2 FP-tree Based Co-location Miner

In this section, an algorithm called FP-CM, for FP-tree based co-location miner is developed based on the framework proposed in section 4.1. FP-tree based co-location miner is an algorithm to generate all and only the co-locations with participation index no less than a user specified participation index threshold min_pi.

FP-tree based co-location miner is described below.

Input:

- A set of spatial features $F = \{1,2,...,k\}$.
- A set of n instances I = {i_1, i_2, …, i_n}, each $i_k \in I$ is a vector <instanceID, spatial feature f, location (x,y)> where $f \in F$ and (x,y) $\in$ spatial framework S.
- A neighbor relationship R over locations in S.
- Participation index threshold min_pi.

Output:

- Co-locations with participation index no less than participation index threshold min_pi.

Algorithm FP-CM

1: $(TD_1, TD_2, ..., TD_k) \leftarrow$ transactionize $(SD_1, SD_2, ..., SD_k)$;
2: *for i = 1 to k do*
3: $MFP_i \leftarrow FP\_MFP(TD_i, \min\_pi)$;
4: *end for*
5: $i \leftarrow 1$;
6: $C \leftarrow \phi$;
7: $C_1 \leftarrow \{1,2,...,k\}$;
8: *while* $C_i \neq \phi$ do
9: $C \leftarrow C \cup C_i$;
10: $C_{i+1} \leftarrow apriori\_gen(C_i)$; // refer to [1]
11: $C_{i+1} \leftarrow prune(C_{i+1}, MFP)$;
12: $i \leftarrow i + 1$;
13: *end while*
14: *C* $\leftarrow$ *multi-way-spatial-join-filter(C, min_pi);*
15: *return C;*

FP-CM algorithm consists of four components, which are transactionization, maximal frequent pattern mining, combining patterns, and pattern filter.

The detailed steps of FP-CM algorithm are explained in the following section 4.2.1, section 4.2.2, section 4.2.3, and section 4.2.4. A spatial dataset as shown in figure 4.2 is used as a running example to make the algorithm easy to understand. This dataset has 8 features, their numbers of instances are specified in table 4.1, and their instances are plotted in a 100 × 100 spatial framework. The neighbor relationship *R* is 12 and the participation index threshold min_pi is 1/3 in this example.

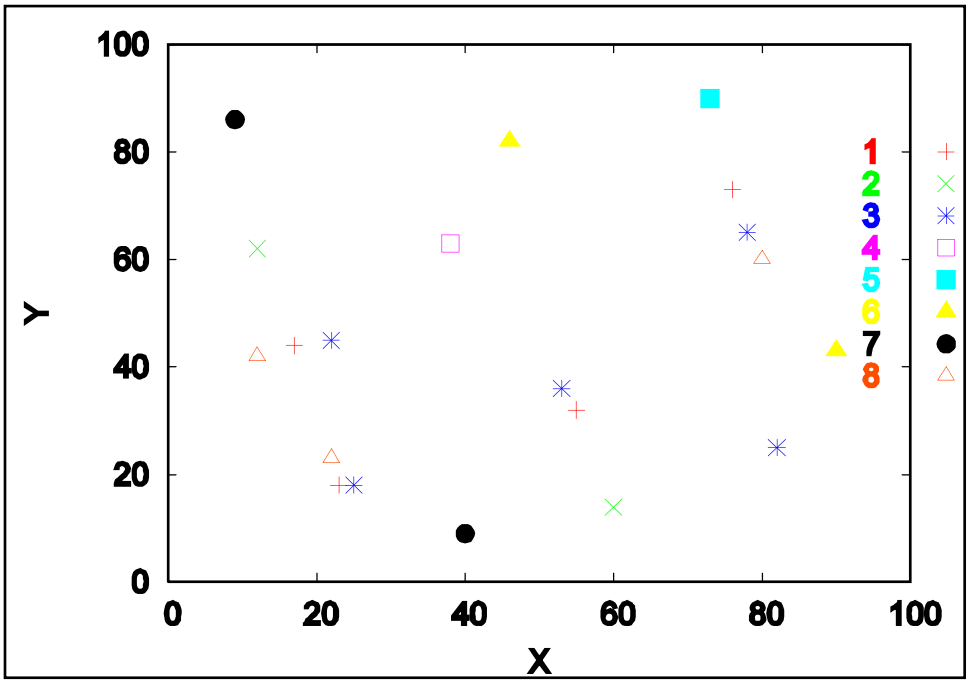


Figure 4.2 Running Example

4.2.1 Transactionization (step 1)

In the transactionization step, reference feature centric model [10], which is reviewed in section 3.2, is used to create transactions for each feature. This step

only needs one database scan. For each spatial feature i, a transaction database TD_i is created as follows. For each instance *o* of feature i, a transaction contains all other spatial features whose instance(s) is (are) the neighbor(s) of instance *o*.

For example, in figure 4.3, spatial feature 1 has four instances. A circle of radius *R* centered at each instance of 1 is drawn as specified in Figure 4.3 and all the other spatial features in each circle are collected. A transaction database with four transactions $TD_1 = \{\{3,8\},\{3,8\},\{3\},\{3\}\}$ is created, which is shown in table 4.1.

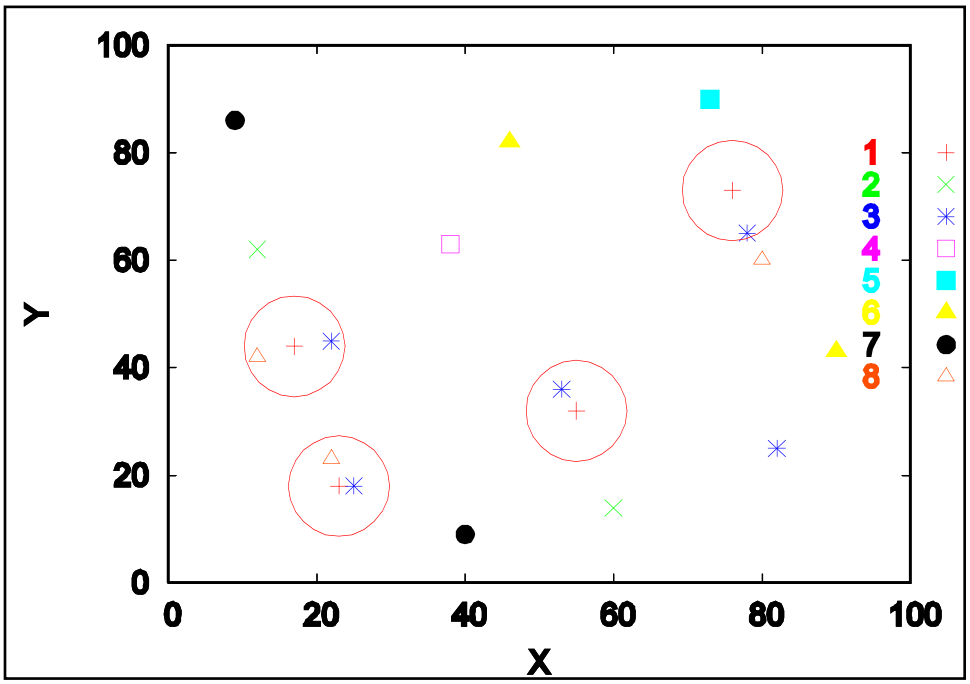


Figure 4.3 Creating Feature 1's Transaction Database

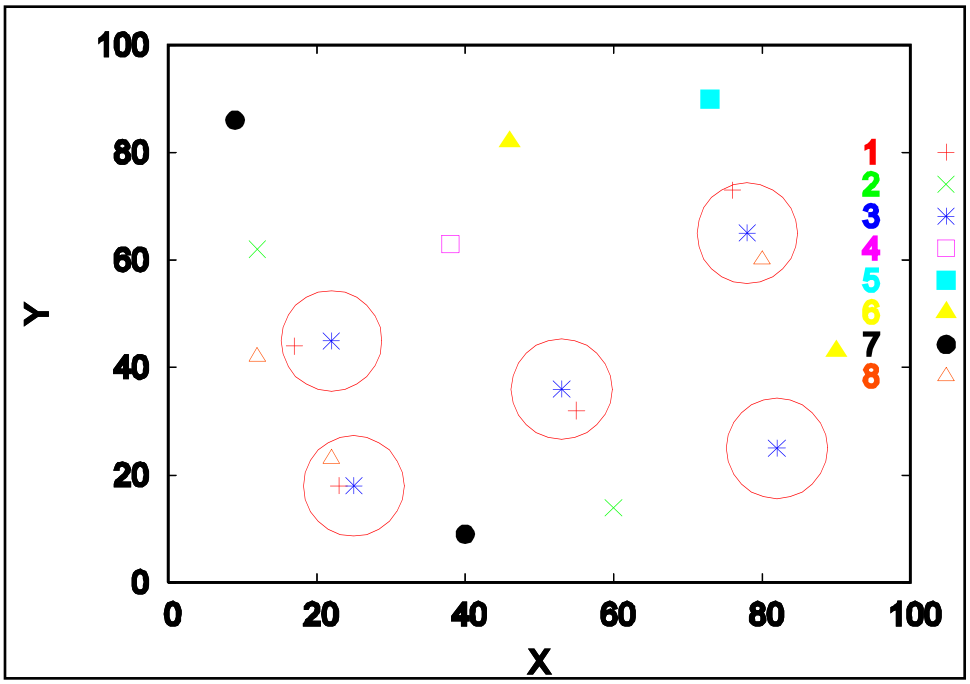


Figure 4.4 Creating Feature 3's Transaction Database

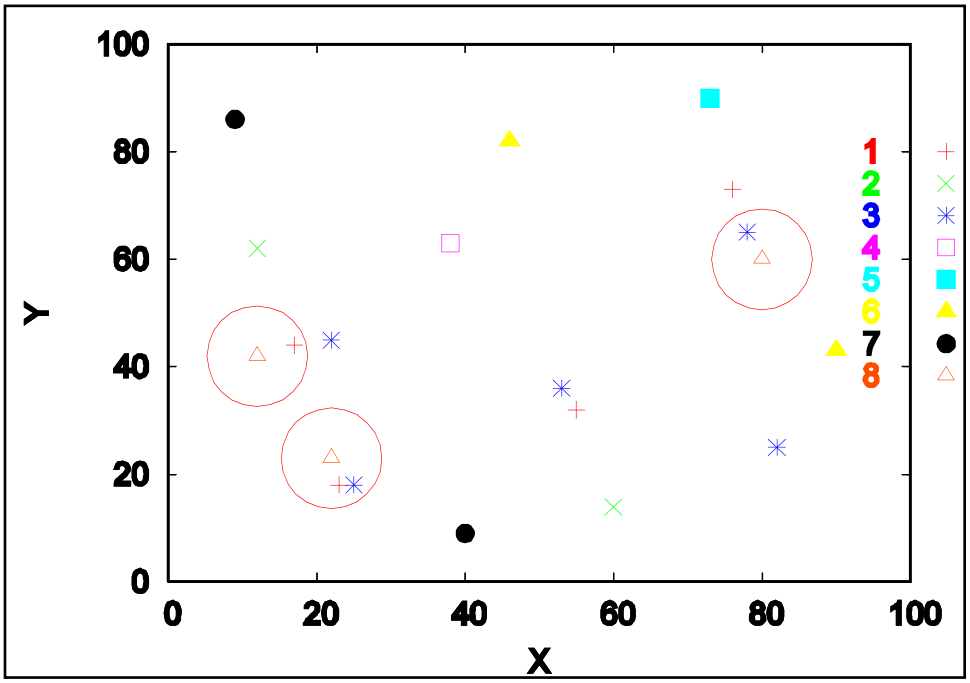


Figure 4.5 Creating Feature 8's Transaction Database

Similarly, in figure 4.4 and 4.5, transaction databases are created for feature 3 and 8 as shown in the third columns of table 4.1. Note that some instances of a feature introduce empty transactions because they do not have other features in their neighborhood *R*. Also, because feature 2, 4, 5, 6, and 7 do not have other features in their neighborhood *R*, their transaction databases are empty.

Feature	# of Instances	Transaction Database TD_i	MFP_i (min_pi = 1/3)
1	4	{3,8},{3,8},{3},{3}	{3,8}
2	2		
3	5	{1},{1,8},{1},{1,8}	{1,8}
4	1		
5	1		
6	2		
7	2		
8	3	{1},{1,3},{3}	{1,3}

Table 4.1 Feature, # of Instances, Transaction Database, and MFP_i

4.2.2 Maximal Frequent Pattern Mining (step 2, 3 and 4)

Because the maximal frequent pattern mining step is FP-tree based [6], which is reviewed in chapter 2, it only need two database scans to mine MFPs. For each transaction database TD_i, all maximal frequent patterns MFP_i are found using the FP-tree based maximal frequent pattern mining algorithms [6, 21] using a support threshold min_sup_i. The min_sup_i (support threshold) for frequent patterns of TD_i is equal to min_pi (participation index threshold) for co-location patterns.

For example, in this running example, min_pi is 1/3. Maximal frequent patterns MFP_i mined of each feature are listed in the third column of table 4.1.

Feature 1's MFP_1 is {{3, 8}}, Feature 3's MFP_3 is {{1, 8}}, and Feature 8's MFP_8 is {{1, 3}} as shown in table 4.1.

4.2.3 Combining Patterns (step 5-13)

The basic structure of the combining pattern step is the level-wise structure of co-location miner [16]. However, it does not require expensive spatial joins to calculate participation indexes. Instead, it consults the MFP, which is $\{MFP_1, MFP_2, \ldots, MFP_k\}$, to prune majority of the false candidate co-location patterns. Because this step involves only spatial features and their maximal frequent patterns and usually is a main memory based step, no database scan is needed. This step will produce a superset of the true co-location patterns to feed into the next pattern filter step.

Step 10 generates candidate co-locations C_{i+1}, which is an adoption of the apriori_gen algorithm of the apriori [1] and needs two steps. Suppose the features in C_i are listed in an order.

Generate Candidate C_{i+1}

Step 1: self-joining C_i
insert into C_{i+1}
select $p.f_1, p.f_2, \ldots, p.f_i, q.f_i$
from C_i p, C_i q
where $p.f_1 = q.f_1, \ldots, p.f_{i-1} = q.f_{i-1}, p.f_i < q.f_i$
Step 2: pruning
for each co-location $c \in C_{i+1}$ do
for each size i subset s of c do
if $c \in C_i$
delete c from C_{i+1}

Example of candidate generation: Let C_2 be {{1 2}, {1 3}, {1 4}, {1 5}, {2 3}}. Self-joining: $C_2 \triangleright\triangleleft C_2$, C_3 will be {{1 2 3}, {1 4 5}}. The prune step will delete {1 4 5} because {4 5} is not contained in C_2. Then C_3 will be {{1 2 3}}.

The prune step (step 11) works as follows:

for each $c \in C_{i+1}$
  for each feature $f \in c$
    if $\neg(\exists mfp \in MFP_f \wedge mfp \supseteq (c - f))$
      delete c from C_{i+1}.

For each candidate co-location pattern $c \in C_{i+1}$, if there exists a feature $f \in c$ and MFP_f does not contain a superset of $(c - f)$, then c is pruned. For example, in table 4.1, for candidate co-location $\{1,2\}$, because MFP_1 does not includes a superset of $\{2\},\{1,2\}$ is pruned from the candidate co-location set. For $\{1,3\}$, because MFP_1 includes a superset of {3} and MFP_3 includes a superset of $\{1\},\{1,3\}$ remains in the candidate co-location set. Similarly, we can get $\{1,8\}$ is also a candidate co-location.

From the size 2 candidate co-locations $\{1,3\}$ and $\{1,8\}$, we can generate a candidate $\{1,3,8\}$. Because MFP_1 includes a superset of $\{3,8\}$, MFP_3 includes a superset of $\{1,8\}$ and MFP_8 includes a superset of $\{1,3\}$, $\{1,3,8\}$ is kept in the size 3 candidate co-location set. Similarly, we can get all the candidate co-locations as shown in table 4.2.

Size	Candidate Co-locations After Combining Patterns Step
2	{1,3},{1,8},{3,8}
3	{1,3,8}

Table 4.2 Candidate Co-locations after Combining Patterns Step

This combining patterns step will not falsely delete any true co-location pattern due to the following Lemma.

Lemma 1: If X is a co-location pattern, $\forall f \in X$, MFP_f contains a superset of $(X - f)$, then the combining patterns step will find *X* as a candidate co-location pattern.

Proof: If X is a co-location pattern, then $pi(X) \geq \min\_ pi$, which means $\forall f \in X$, the participation ratio $pr(X, f) \geq \min\_ pi$. This implies that at least min_pi of *f*'s instances participate in any instance of X. Because an instance of X is a clique of X, we can infer that at least min_pi of *f*'s instances have instances of $(X - f)$ in their neighborhood R. So, at least min_sup_f of *f*'s instances contain $(X - f)$ which means $(X - f)$ is a frequent pattern reference to feature f using min_sup_f = min_pi. Since MFP_f contains all the maximal frequent patterns, MFP_f contains a superset of $(X - f)$. Therefore, the combining patterns steps will find X as a candidate co-location pattern.

This Lemma establishes the relationship between $pr(X, f)$ (the participation ratio of a feature f in a pattern X) and $(X - f)'s$ presence in the maximal frequent pattern set of feature *f* using min_sup_f = min_pi. Once we have all the maximal frequent pattern sets of all the spatial features, this property will help us in judging whether a pattern X is a candidate co-location pattern by checking whether $(X - f)$ is included in at least one maximal frequent pattern of MFP_f for $\forall f \in X$.

4.2.4 Pattern Filter (step 14 and 15)

Once the total number of candidate co-location patterns are reduced from $2^{\#\_of\_features}$ to a small superset of the true co-location patterns, a hash based spatial join technique [7] and a multi-way spatial joins [11] approach are used to filter out the false co-location patterns. Spatial datasets are hashed into buckets using a grid [7] and then a multi-way spatial join, which is based on a backtracking search heuristic [11], is used to find all the maximal cliques. The list of all the candidate co-location patterns from the previous steps is kept and the cliques are registered to their corresponding candidate co-location patterns. Finally, the participation index for each candidate co-location pattern is calculated and the set of all true co-location patterns is returned. As shown in the following, the pattern filter step includes a hashing step and a mining step and each step only needs one database scan.

Hashing Step
1: impose a regular grid over the map;
2: for each feature *i*
3: each instance of *i* is extended by distance *R* to form a disk and hashed into the cells intersected by this disk;

Mining Step
4: for each cell C of the grid
5: for each feature *i* in C
6: load bucket SD_i^c in memory; // SD_i^c is the instances in SD_i hashed //into cell C
7: sort instances of SD_i^c in order based on their x-values;
8: for each feature *i* in C
9: for each instance *o* of SD_i^c
10: find the instances of other features that are close to instance o;
11: if (the above instances of other features are close to each other ^the set of these features is in the list of candidate co-locations)
12: keep clique pattern instances where instance *o* participates in;
13: for each clique pattern C
14: compute the participation ratios of features in C and get pi(C);

15: if (pi(C) >= min_pi)
16: C is a co-location pattern;

For example, after getting the pi of each candidate co-location in table 4.2, we can verify whether they are co-locations as shown in table 4.3. Each pi of size 2 candidate co-locations satisfies min_pi, so they are all co-locations. Because pi ({1,2,8}) is less than min_pi, {1,3,8} is not a co-location.

Size	Candidate Co-locations	Instances	Participation Index	Co-location? (min_pi=1/3)
2	{1,3}	{1,1},{2,2},{3,3},{4,4}	min{4/4,4/5}=4/5	Yes
	{1,8}	{1,1},{2,2}	min{2/4,2/3}=2/3	Yes
	{3,8}	{2,2},{4,3}	min{2/5,2/3}=2/5	Yes
3	{1,3,8}	{2,2,2}	min{1/4,1/5,1/3}=1/5	No

Table 4.3 Size, Candidate Co-locations, Instances, and Participation Index

4.3 Summary

In this chapter, a FP-tree based co-location mining framework is propoed, which is composed of four function modules (transactionization, maximal frequent pattern mining, combining patterns, and pattern filter). An algorithm called FP-CM for FP-tree based co-location miner is also proposed based on the proposed framework. Because transactionization step needs one database scan, maximal frequent pattern mining step requires two database scans, combining patterns step, which only involves the spatial features and their maximal frequent pattern sets, is a main memory based step, and pattern filter step needs two database scans, the total number of database scans of FP-CM algorithm is bound by 5. The detailed steps of the FP-CM algorithm are explained using a running example to make it clear.

CHAPTER 5

ALGORITHM ANALYSIS AND EXPERIMENTAL EVALUATION

A FP-tree based co-location mining framework is proposed, a FP-tree based co-location miner algorithm FP-CM is also proposed based on the proposed framework and the detailed steps of the proposed algorithm are explained using a running example in the previous chapter. This chapter analyzes and evaluates FP-CM algorithm in comparison with the previous work. Section 5.1 shows it satisfy completeness, correctness and computational efficiency which are the essential requirements for spatial co-location pattern mining. To show the performance comparison of FP-CM with generation-and-test based algorithm CM, both algorithms are implemented and a series of experiments are systematically conducted for them. The experimental results show that FP-CM algorithm outperforms CM algorithm by an order of magnitude. The experimental evaluation is presented in section 5.2.

5.1 Algorithm Analysis

This section will present an analysis of proposed FP-tree based co-location Miner algorithm for its completeness, correctness, and computational efficiency.

5.1.1 Completeness

FP-tree based co-location miner algorithm is complete if it finds all spatial co-locations with participation index no less than participation index threshold.

Lemma 2: FP-tree based co-location miner algorithm is complete. Any co-location pattern C with pi(C) $\geq$ min_pi will be found by FP-CM algorithm.

Proof:

Step 1 uses reference feature centric model to create a transaction database for each feature. It is complete.

Step 2, 3 and 4 can find all maximal frequent patterns MFP_i using the FP-tree based maximal frequent pattern mining algorithms using a support threshold min_sup_i equal to min_pi (participation index threshold) for co-location patterns for each transaction database TD_i because of the completeness of the FP-tree based maximal frequent pattern mining algorithms [6, 21].

Step 5-13 consult the MFP, which is $\{MFP_1, MFP_2, \ldots, MFP_k\}$, to prune majority of the false candidate co-location patterns to produce a superset of the true co-location patterns. The loop from step 8 to 13 iterates to generate candidate co-locations of size 2 or more. The self-joining C_i of step 10 produces candidates C_{i+1}, the pruning C_{i+1} of step 10 only prunes candidates whose at least one subset is not in C_i and the pruned candidates must not be prevalent co-locations due to the monotonic non-increasing of the participation index. So, step 10 is complete. Because of

Lemma 1, which is proved in chapter 4, the pruning (step 11) using the maximal frequent pattern sets does not eliminate any true co-location pattern. So, step 11 is complete.

Step 14 and 15 can find all the co-locations because the multi-way spatial join algorithm correctly calculates the participation indexes of candidate co-locations which are a superset of the true co-location patterns and generated by the previous steps, the participation indexes are used to prune the false candidate co-locations, and any pruned candidate co-location must not be prevalent due to the correct calculation of the participation indexes of the multi-way spatial join algorithm.

5.1.2 Correctness

FP-tree based co-location miner algorithm is correct if any spatial co-location it finds has participation index no less than participation index threshold.

Lemma 3: FP-tree based co-location miner is correct. Any co-location pattern C found by FP-CM algorithm has pi(C) $\geq$ min_pi.

Proof:

Step 1 is correct because using reference feature centric model can correctly create a transaction database for each feature.

Step 2, 3 and 4 can find only maximal frequent patterns MFP_i using the FP-tree based maximal frequent pattern mining algorithms using a support threshold min_sup_i equal to min_pi (participation index threshold) for co-location patterns for each transaction database TD_i because of the

correctness of the FP-tree based maximal frequent pattern mining algorithms [6, 21].

Step 5-13 consult the MFP, which is {$MFP_1, MFP_2, \ldots, MFP_k$}, to prune majority of the false candidate co-location patterns to produce a superset of the true co-location patterns. The loop from step 8 to 13 iterates to generate candidate co-locations of size 2 or more. Step 10 is correct because the self-joining C_i of step 10 correctly produces candidates C_{i+1} and the pruning C_{i+1} of step 10 only prunes candidates whose at least one subset is not in C_i. Step 11 produces a superset of the true co-location patterns using the maximal frequent pattern sets correctly. So, step 11 is correct.

Step 14 and 15 are correct because the multi-way spatial join algorithm correctly calculates the participation indexes, which are used to filter out the false candidate co-locations, and generates only co-locations having pi(C) $\geq$ min_pi.

5.1.3 Computational Efficiency

Computational efficiency means IO and CPU cost to find all and only co-locations should be acceptable.

The proposed FP-tree based co-location miner algorithm only requires a small number of database scans. One database scan is required in the transactionization step for creating transactions for each spatial feature in the spatial data. The maximal frequent pattern mining step requires two database

scans because it is FP-tree based. No database scan is needed in the combining patterns step because it involves only spatial features and their maximal frequent patterns and usually is a main memory based step. Finally, the pattern filter step using grid and multi-way spatial joins requires two more database scans. So the total number of database scans of FP-CM algorithm is bounded by a small constant number 5, which guarantees the computational efficiency of the proposed FP-CM algorithm and proves it is much faster in comparison with the candidate generation-and-test spatial co-location miner algorithm [16] which needs many database scans.

5.2 Experimental Evaluation

5.2.1 System Settings

To evaluate the performance of FP-CM, co-location miner (CM) [16] is chosen for comparison. Both FP-CM and CM are implemented using C++ languages. All the experiments are carried out on a 2.40GHz Pentium IV PC with 1G main memory, running on Debian linux operating system

5.2.2 Synthetic Data Generation

A synthetic dataset generator is designed for evaluation. First, a set of P maximal co-locations is pre-generate similar as that in [1]. Then, |I| instances are generated. Each time an instance of a maximal co-location pattern is generated with (prob) probability and a random instance is generated with (1-prob) probability. In the first case, a maximal co-location is chose from the set of P pre-

generated maximal co-locations. Then, a square of size $N_d \times N_d$ is chose from the space framework of size $S_d \times S_d$, and an instance for each feature in the chosen maximal co-location is generated and put uniformly into the square. In the second case, a feature is chose with equal probability from the feature set NF and an instance of that feature is put uniformly into the space. The following table 6.1 lists all of the parameters used in the data generator and experiments.

Parameter	Definition
P	Potential maximal co-location patterns
PS	Size of maximal potential co-location patterns
F	Spatial feature set that participates in pattern formation
NF	Noise spatial feature set
I	All Spatial instances
N_d	Neighborhood size of the square
S_d	Spatial framework size
prob	probability of an instance to be in a co-location pattern
min_pi	participation index threshold

Table 5.1 Parameters in the Data Generator to Generate the Synthetic Data

5.2.3 Experimental Results

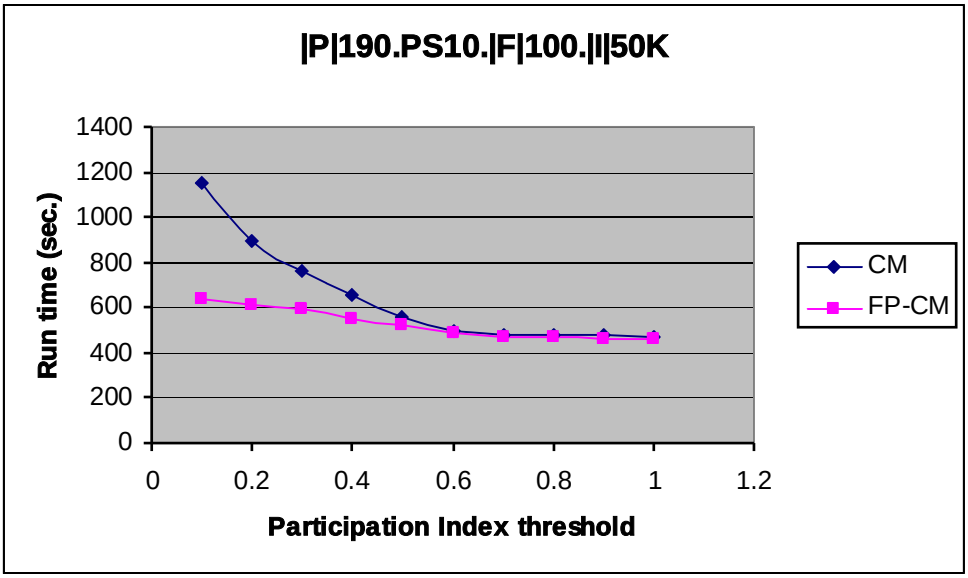
A series of experiments are systematically conducted. The experiments are extensive and the results are consistent. Limited by space, only some representative experimental results for various parameters are reported. The run time used in the following figures means the total execution time, which is the period between input and output, instead of CPU time.

For simplicity, experiments are named using the parameters in Table 5.1. For example, |P|50.PS5.|F|250.|I|50K denotes an experiment with 50 pre-generated maximal co-location patterns whose average size is 5, 250 features participating in pattern formation, and 24K spatial instances. All the results reported have N_d = 10, S_d =1000, and prob=80%.

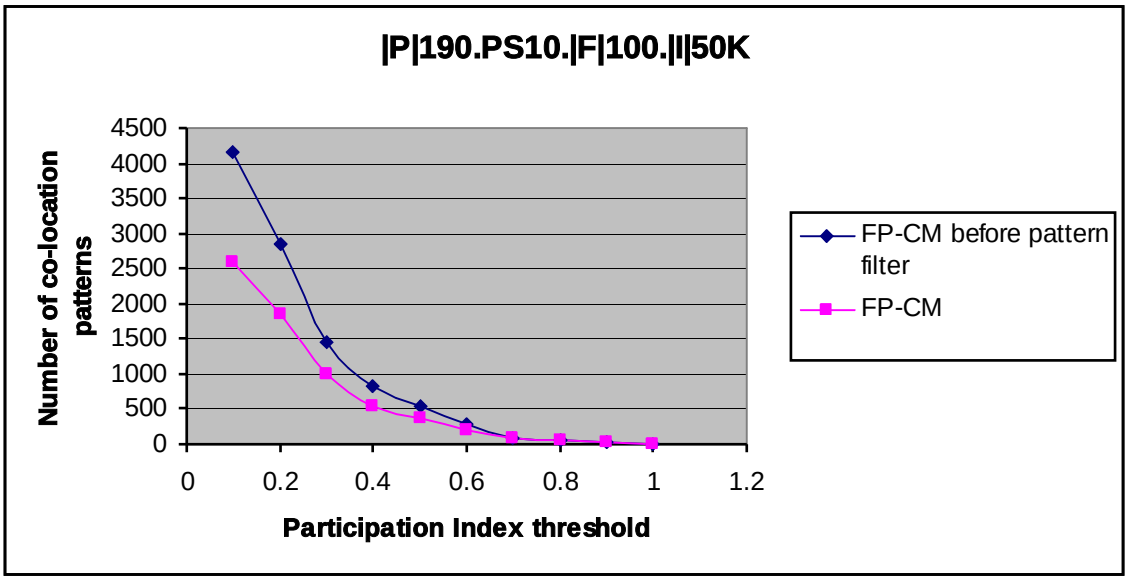
5.2.3.1 Effect of Participation Index Threshold

The performance of the two algorithms with respect to the participation index threshold is shown in Figure 5.1(a). Clearly, FP-CM is much more scalable than CM. This is because as the participation index threshold decreases from 1 to 0.1, the number as well as the length of co-location patterns increases dramatically. The candidates and associated spatial joins that CM has to handle becomes extremely large, therefore, CM becomes very slow when the participation index threshold is low. This is also because the bulk of FP-CM run time is the FP-tree based maximal frequent pattern generation and this is usually a constant number of database scans. CM is more sensitive to the larger spatial dataset since its number of database scans is not bounded by a small constant.

The difference between the number of candidate co-location patterns generated by FP-CM before pattern filter and the number of co-location patterns generated by FP-CM after pattern filter becomes larger and larger as the participation index threshold goes down as shown in Figure 5.1(b).



(a)

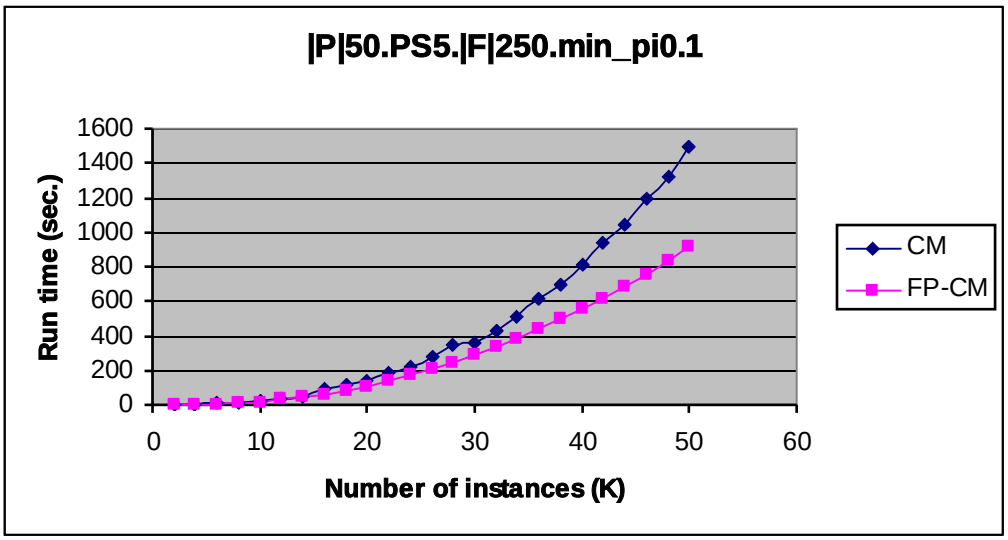


(b)

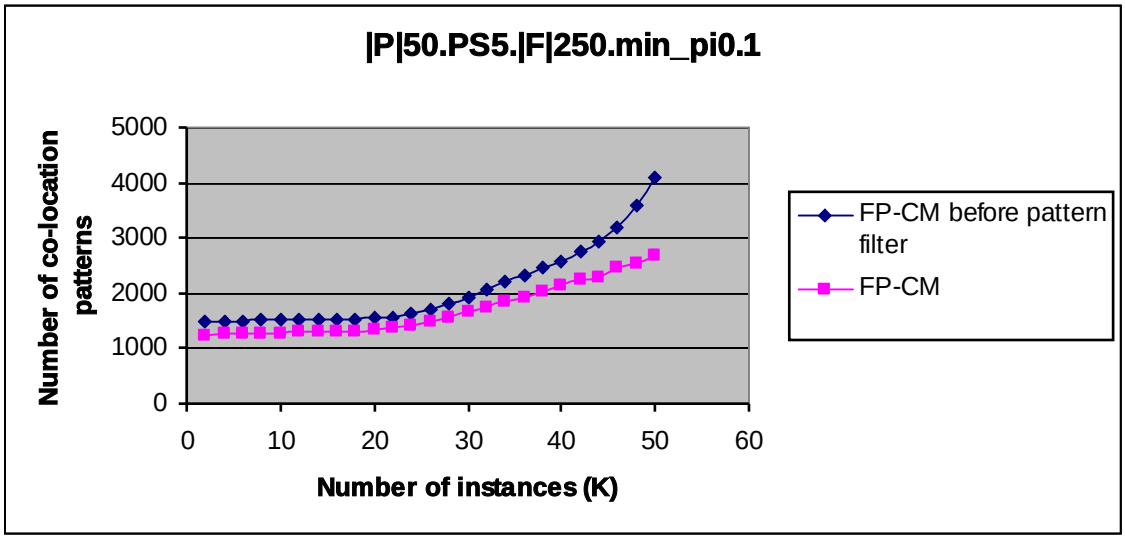
Figure 5.1 Effect of Participation Index Threshold

5.2.3.2 Effect of Number of Instances

The scalability of the two algorithms as the number of instances increases from 2K to 50K is shown in Figure 5.2(a). FP-CM scales much better than CM. This is because as the number of instances grows up, the number as well as the length of co-location patterns increases. The candidates and associated spatial joins that CM has to handle becomes large, therefore, CM becomes slow when the number of instances is high. This is also because the bulk of FP-CM run time is the FP-tree based maximal frequent pattern generation and this is usually a constant number of database scans. CM is more sensitive to the larger spatial dataset since its number of database scans is not bounded by a small constant. The difference between the number of candidate co-location patterns generated by FP-CM before pattern filter and the number of co-location patterns generated by FP-CM after pattern filter becomes larger and larger as the number of instances increases from 2K to 50K, which is shown in Figure 5.2(b).



(a)



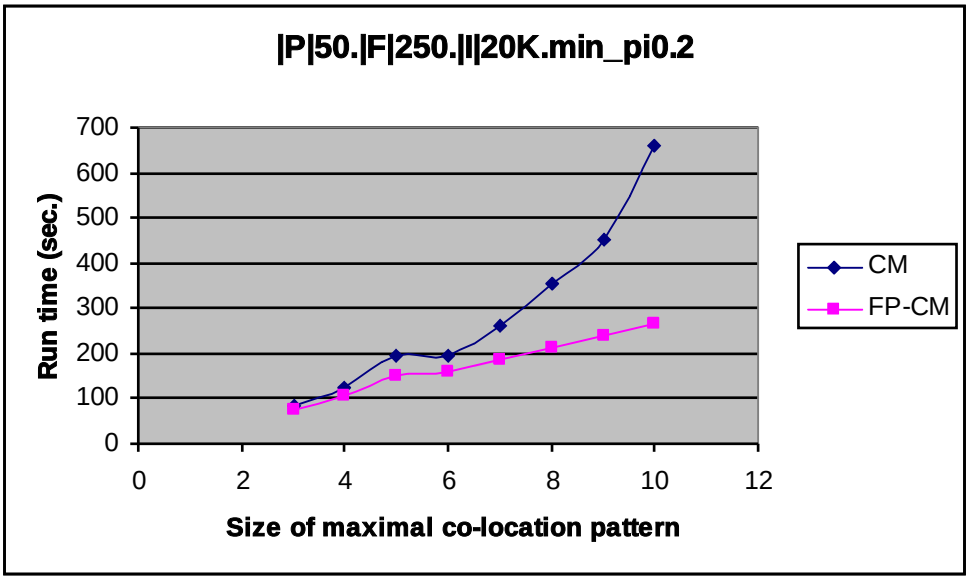
(b)

Figure 5.2 Effect of Number of Instances

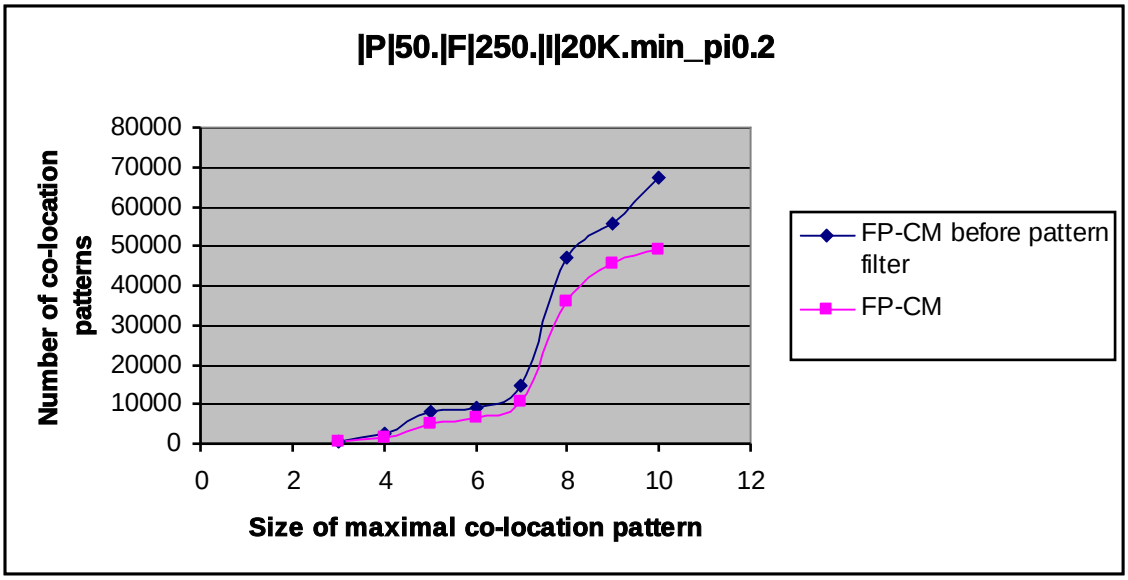
5.2.3.3 Effect of Size of Maximal Co-location Pattern

Figure 5.3(a) shows FP-CM scales much better than CM when the size of maximal co-location pattern ranges from 3 to 10. This is because as the size of maximal co-location pattern grows up, the number as well as the length of co-location patterns increases. The candidates and associated spatial joins that CM has to handle becomes large, therefore, CM becomes slow when the size of maximal co-location pattern is high. This is also because the bulk of FP-CM run time is the FP-tree based maximal frequent pattern generation and this is usually a constant number of database scans. CM is more sensitive to the larger spatial dataset since its number of database scans is not bounded by a small constant. The difference between the number of candidate co-location patterns generated by FP-CM before pattern filter and the number of co-location patterns generated

by FP-CM after pattern filter becomes larger and larger as the size of maximal co-location pattern increases from 3 to 10, which is shown in Figure 5.3(b).



(a)

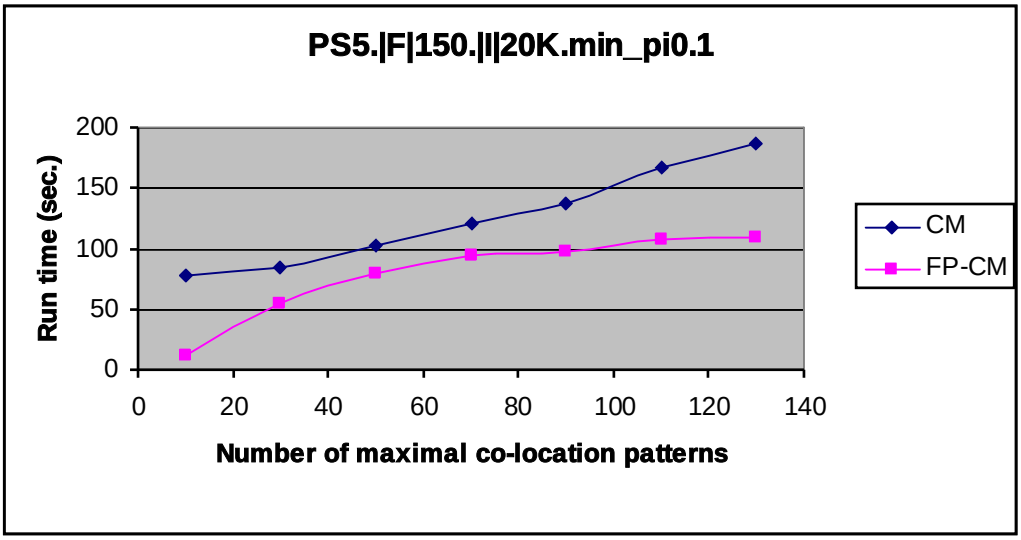


(b)

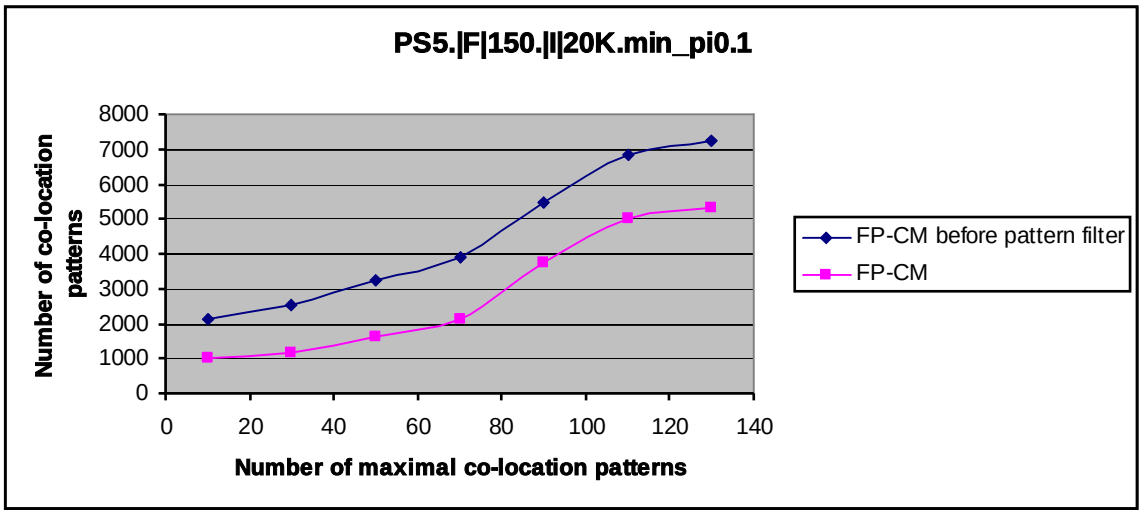
Figure 5.3 Effect of Size of Maximal Co-location Patterns

5.2.3.4 Effect of Number of Maximal Co-location Patterns

Figure 5.4(a) shows the performance comparison of the two algorithms as the number of maximal co-location patterns increases from 10 to 130. FP-CM scales much better than CM. This is because as the number of maximal co-location patterns grows up, the number as well as the length of co-location patterns increases. The candidates and associated spatial joins that CM has to handle becomes large, therefore, CM becomes much slow when the number of maximal co-location patterns is high. This is also because the bulk of FP-CM run time is the FP-tree based maximal frequent pattern generation and this is usually a constant number of database scans. CM is more sensitive to the larger spatial dataset since its number of database scans is not bounded by a small constant. The difference between the number of candidate co-location patterns generated by FP-CM before pattern filter and the number of co-location patterns generated by FP-CM after pattern filter becomes larger and larger as the number of maximal co-location patterns increases from 10 to 130, which is shown in Figure 5.4(b).



(a)



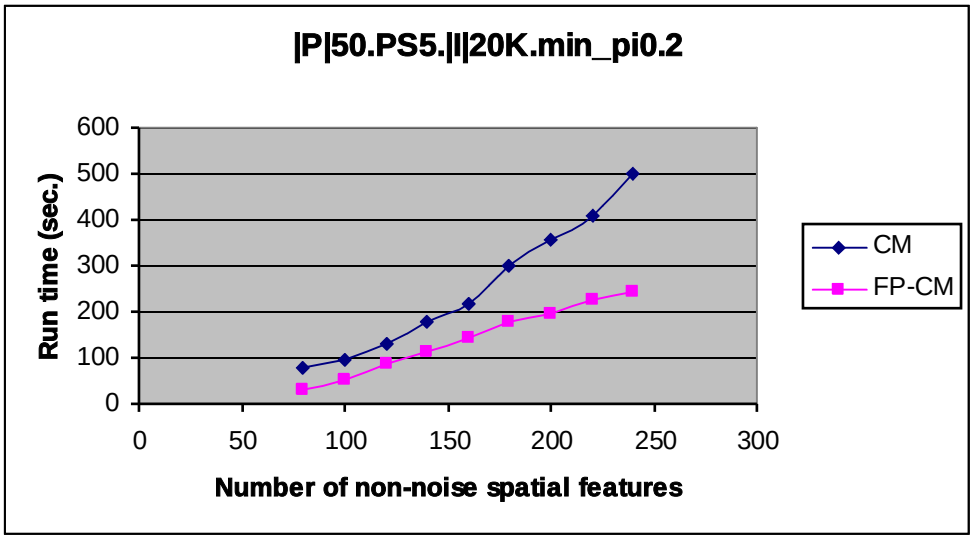
(b)

Figure 5.4 Effect of Number of Maximal Co-location Patterns

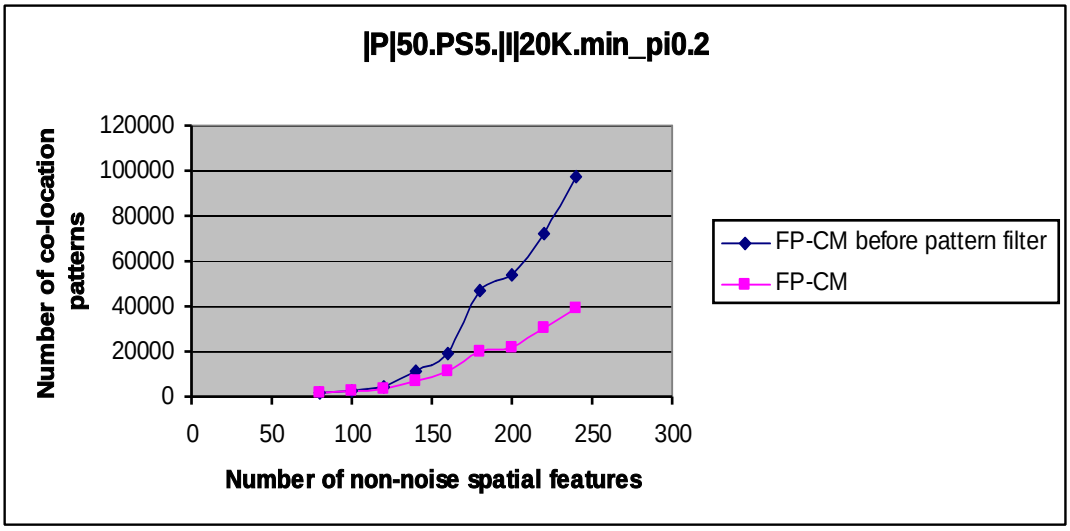
5.2.3.5 Effect of Number of Non-noise Spatial Features

The performance of the two algorithms with respect to the number of features that participates in pattern formation (non-noise features) is shown in Figure 5.5(a). FP-CM scales much better than CM. as the number of non-noise

spatial features increases from 60 to 240. This is because as the number of non-noise spatial features grows up, the number and the length of co-location patterns increase. The candidates and associated spatial joins that CM has to handle becomes large, therefore, CM becomes slow when the number of non-noise features is high. This is also because the bulk of FP-CM run time is the FP-tree based maximal frequent pattern generation and this is usually a constant number of database scans. CM is more sensitive to the larger spatial dataset since its number of database scans is not bounded by a small constant. The difference between the number of candidate co-location patterns generated by FP-CM before pattern filter and the number of co-location patterns generated by FP-CM after pattern filter becomes larger and larger as number of non-noise features increases shown in Figure 5.5(b).



(a)



(b)

Figure 5.5 Effect of Number of Non-noise Spatial Features

5.3 Summary

In this chapter, completeness, correctness and computational efficiency of the proposed FP-tree Based Co-location Miner FP-CM algorithm are proved. The performance comparison of FP-CM algorithm with the candidate generation-and-test based spatial co-location pattern miner CM algorithm are also shown. The extensive experimental results show that FP-CM algorithm outperforms CM algorithm by an order of magnitude. Therefore, the algorithm analysis and experimental evaluation both show that the proposed FP-CM algorithm outperforms CM algorithm.

CHAPTER 6

CONCLUSION

Since its introduction, the paradigm of frequent pattern mining has undergone a shift from candidate generation-and-test based frequent pattern mining to projection based frequent pattern mining. A finite set of disjoint transactions is crucial in frequent pattern definition and its mining algorithms. Spatial co-location patterns resemble frequent patterns in many aspects. However, the difficulty in creating explicit disjoint transactions from continuous spatial data makes the similar paradigm shift in spatial co-location pattern mining very difficult. In this thesis, a candidate generation-and-test based frequent pattern mining, a FP-tree (projection) based frequent pattern mining paradigm, a candidate generation-and-test based spatial co-location pattern mining paradigm, and a multi-way spatial joins based co-location pattern mining paradigm are investigated. In particular, a FP-tree based co-location mining framework is proposed, which is flexible in incorporating any fast maximal frequent pattern mining algorithm developed in literature to help spatial co-location pattern mining. A FP-tree based co-location miner FP-CM algorithm based on the proposed FP-tree based co-location mining framework is also proposed. It combines the salient features of FP-tree based maximal frequent pattern mining [6] and fast

multi-way spatial joins [11] to reduce the total number of database scans to a small constant number. It is proved that FP-CM algorithm is complete, correct, and only requires a small constant number of database scans. The extensive experimental results also show that FP-CM algorithm is an order of magnitude faster than the candidate generation-and-test based spatial co-location miner CM algorithm. Thus, the algorithm analysis and experimental evaluation both show that the proposed FP-CM algorithm outperforms CM algorithm.

The proposed FP-tree based co-location mining framework could be treated as a new data-driver partitioning of spatial datasets according to the instances of each spatial feature. Compared with traditional spatial partition approaches [20], this approach does not have the problem of combinatorial explosion of temporary candidate patterns needed to maintain by the algorithm before all the partitions are processed as acknowledged by the authors in [20]. In future work, comparing various partition based co-location pattern mining algorithms would be an interesting and imperative research direction.

BIBLIOGRAPHY

1. Rahesh Agrawal and Ramakrishnan Srikant. Fast Algorithms for Mining Association Rules. In International Conference on Very Large Data Bases (VLDB'94), Santiago de Chile, 1994, pp 487-499.
2. N. A. C. Cressie. Statistics for Spatial Data. Wiley and Sons, ISBN: 0471843369, 1991.
3. M. Ester, A. Frommelt, H. P. Kriegel, and J. Sander. Algorithms for Characterization and Trend Detection in Spatial Databases. In Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York City, August 27-31, 1998.
4. V. Estivill-Castro and I. Lee. Data Mining Techniques for Autonomous Exploration of Large Volumes of Geo-referenced Crime Data. In Proc. of the 6th International Conference on Geocomputation, Brisbane, Australia, September, 2001.
5. V. Estivill-Castro and A. Murray. Discovering Associations in Spatial Data – An Efficient Medoid Based Approach. In Proc. of the Second Pacific-Asia Conference on Knowledge Discovery and Data Mining, Melbourne, Australia, April, 1998.

6. Jiawei Han, Jian Pei, and Yiwen Yin. Mining Frequent Patterns without Candidate Generation. In Proc. of ACM SIGMOD International Conference on Management of Data, Dallas, USA, May, 2000, pp 1-12.
7. Jignesh M. Patel and David J. DeWitt. Partition Based Spatial-Merge Join. In Proc. of ACM SIGMOD International Conference on Management of Data, Montreal, Quebec, Canada, June 4-6, 1996.
8. E. M. Knorr, R. T. Ng, and D. L. Shilvock. Finding Boundary Shape Matching Relationships in Spatial Data. In Proc. 5th International Symposium on Large Spatial Database, Berlin, Germany, July, 1997, pp 29-46.
9. K. Koperski, and J. Adhikary, and J. Han. Spatial Data Mining: Progress and Challenges. In SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD'96), Montreal, Canada, 1996.
10. K. Koperski and Jiawei Han. Discovering of Spatial Association Rules in Geographic Information Databases. In Proc. 4th International Symposium on Large Spatial Database, Portland, ME, 1995, pp47-66.
11. Nikos Mamoulis and Dimitris Papadias. Multiway Spatial Joins. ACM Trans. Database Syst. 26(4):424—475, 2001.
12. Yasuhiko Morimoto. Mining Frequent Neighboring Class Sets in SpatialDatabases. In Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August, 2001.

13. R. T. Ng and J. Han. Efficient and Effective Clustering Methods for Spatial Data Mining. In Proc. 20th International Conference on Very Large Data Bases. Santiago, Chile, 1994, pp 144-155.
14. J. F. Roddick and M. Spiliopoulou. A Bibliography of Temporal, Spatial and Spatio-temporal Data Mining Research. ACM Special Interest Group on Knowledge Discovery and Data Mining Explorations, 1999.
15. S. Shekhar and S. Chawla. Spatial Databases: A Tour. Prentice Hall, ISBN: 0130174807, 2003.
16. Shashi Shekhar and Yan Huang. Discovering Spatial Co-location Patterns: A Summary of Results. In Proc, 7th Intl. Symposium on Spatial and Temporal Databases, Redondo Beach, CA, July, 2001.
17. S. Shekhar, C.T. Lu, and P. Zhang. Detecting Graph-based Spatial Outliers: Algorithms and Applications. In Proc, 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, 2001.
18. S. Shekhar, P. Schrater, W.R. Raju, W. Wu, and S. Chawla. Spatial Contextual Classification and Prediction Models for Mining Geospatial Data. IEEE Transactions on Multimedia (Special Issue on Multimedia Databases), 2002.
19. W. Wang, J. Yang, and R. Muntz. STING: A Statistical Information Grid Approach to Spatial Data Mining. In International Conference on Very Large Data Bases, Athens, Greece, 1997.

20. Xin Zhang, Nikos Mamoulis, David W. Cheung, and Yutao Shou. Fast Mining of Spatial Collocations. In Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, USA, August, 2004.
21. Gösta Grahne and Jianfei Zhu. Efficiently Using Prefix-trees in Mining Frequent Itemsets. In Proc. of the First IEEE ICDM Workshop on Frequent Itemset Mining Implementations (FIMI'03), Melbourne, FL, November, 2003.
22. L. Arge, O. Procopiuc, S. Ramaswamy, T. Suel, and J. Vitter. Scalable Sweeping-based Spatial Join. In Proc. of the International Conference on Very Large Databases, New York City, 1998.
23. Jiawei Han and Micheline Kamber. Data Mining: Concepts and Techniques. The Morgan Kaufmann Series in Data Management Systems, Jim Gray, Series Editor Morgan Kaufmann Publishers, August, 2000. ISBN: 1-55860-489-8.
24. Sergey Brin, Rajeev Motwani, and Craig Silverstein. Beyond Market Basket: Generalizing Association Rules to Correlations. In Proc. ACM SIGMOD International Conference on Management of Data, Tucson, Arizona, May, 1997, pp. 265-276.
25. Roberto J. Bayardo Jr. Efficiently Mining Long Patterns from Databases. In Proc. ACM SIGMOD International Conference on Management of Data, Seattle, Washington, United States, June, 1998, pp. 85-93.

26. Rakesh Agrawal, Ramakrishnan Srikant. Mining Sequential Patterns. In International Conference on Data Engineering (ICDE'95), Taipei, Taiwan, February, 1995, pp.3-14.

27. Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Mining AssociationRules between Sets of Items in Large Databases. In Proc. of ACM SIGMOD International Conference on Management of Data, Washington DC, 1993, pp 207-216.

28. T. Brinkhoff, H. P. Kriegel, and B. Seeger. Efficient Processing of Spatial Joins Using R-trees. In Proc. of ACM SIGMOD International Conference on Management of Data, Washington DC, 1993.

29. David Hand, Heikki Mannila, and Padhraic Smyth. Principles of Data Mining. MIT Press, ISBN: 026208290X, 2001.

30. Wai-Ki Ching, Michael Kwok-Po Ng. Advances on Data Mining and Modeling. ISBN: 9812383549, 2001.

31. Willi Klösgen and Jan M. Zytkow. Handbook of Data Mining and Knowledge Discovery. Oxford University Press, ISBN: 0195118316, 2001.

32. Roy Ladner, Kevin Shaw, Mahdi Abdelguerfi. Mining Spatio-temporal Information Systems. Kluwer Academic Publishers, ISBN: 1402071701, 2002.

33. Bernhard Hofmann-Wellenhof, H. Lichtenegger, and J. Collins. Global Positioning System: Theory and Practice. Springer-Verlag Wien New York, ISBN: 3211828397, 1997.

34. John Stillwell, Graham Clarke. Applied GIS and Spatial Analysis. Wiley, ISBN: 0470844094, 2004.

35. Tor Bernhardsen. Geographic Information Systems: An Introduction. Wiley, ISBN: 0471419680, 2001.

36. Michael F. Worboys. GIS: A Computing Perspective. Taylor & Francis, ISBN: 0748400648, 1995.

37. Philippe Rigaux, Michel Scholl, Agnès Voisard. Spatial Databases: With Application to GIS. Morgan Kaufmann Publishers, ISBN: 1558605886, 2002.

38. Yue-Hong Chou. Exploring Spatial Analysis in Geographic Information Systems. Onword Press, ISBN: 1566901197, 1997.

39. Robert Haining. Spatial Data Analysis in the Social and Environmental Sciences. Cambridge University Press, ISBN: 0521384168, 1990.

40. Yan Huang, Shashi Shekhar, and Hui Xiong. Discovering Co-location Patterns from Spatial Datasets: A General Approach. IEEE Transactions on Knowledge and Data Engineering (TKDE), VOL. 16, NO. 12, December 2004.

41. M.J. Zaki and K. Gouda. Fast Vertical Mining Using Diffsets. In Proc. of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, August, 2003.

42. R. Munro, S. Chawla, and P. Sun. Complex Spatial Relationships. In Proc. of the 3rd IEEE International Conference on Data Mining (ICDM), Melbourne, Florida, USA, November, 2003.

43. Yan Huang, Hui Xiong, Shashi Shekhar, and Jian Pei. Mining Confident Co-location Rules without a Support Threshold. In Proc. of the 18th ACM Symposium on Applied Computing (ACM SAC), Melbourne, Florida, USA, March, 2003.

44. J. Sander, M. Ester, H. P. Kriegel, and X. Xu. Density-based Clustering in Spatial Databases: A New Algorithm and its Applications. Data Mining and Knowledge Discovery, 2(2):169-194, 1998.

45. S. Guha, R. Rastogi, and K. Shim. CURE: An Efficient Clustering Algorithm for Large Databases. In Proc. of ACM SIGMOD International Conference on Management of Data, New York City, 1998, pp73-84.

46. K. Koperski, J. Han, and N. Stefanovic. An Efficient Two-step Method for Classification of Spatial Data. In Proc. International Symposium on Spatial Data Handling, Vancouver, Canada, 1998.

47. J. Hipp, U. Guntzer, and G. Nakaeizadeh. Algorithms for Association Rule Mining – A General Survey and Comparison. In Proc. ACM SIGKDD International Conference on Knowledge Discovery and DataMining, Boston, MA, USA, 2000.