Approaches towards Co-location Rule Mining

M.Tech Seminar Report

Submitted in partial fulfillment of the requirements for the degree of

Master of Technology

by

Gorijala Mohana Krishna

Roll No: 123059006

under the guidance of

Prof. Umesh Bellur

Department of Computer Science and Engineering
Indian Institute of Technology, Bombay
Mumbai
Acknowledgements

I take this opportunity to thank my guide Prof. Umesh Bellur for his exemplary guidance and constant mentoring throughout the seminar period. I would also like to express my gratitude to Prof. Dhaval Patel of IIT-Roorkee for giving initial inputs and answering all my questions patiently. His help at the beginning of seminar to find a topic is so valuable.
## Contents

1 Introduction .................................................. 1

2 Background ..................................................... 3
   2.1 Spatial Data Mining(SDM) ................................ 3
   2.2 Unique Properties of Spatial Data ..................... 4
   2.3 Co-location Pattern and Rule Mining ................... 4
   2.4 Difference between co-location mining and traditional association rule mining ................................................................. 6
   2.5 Neighborhood Enumeration ................................ 6
   2.6 Problem Statement ......................................... 8
   2.7 Other Basic Concepts ..................................... 9
      2.7.1 Definitions ........................................... 9
      2.7.2 Interestingness Measures ........................... 9

3 Various approaches for Co-location rule Mining and their analysis 11
   3.1 Brute-force Approach: .................................... 11
   3.2 Posing as Graph Problem: ............................... 12
   3.3 Join-Based Approach .................................... 12
      3.3.1 Analysis ............................................. 17
   3.4 Partial-Join Approach .................................. 17
      3.4.1 Analysis ............................................. 21
   3.5 Join-Less Approach .................................... 22
      3.5.1 Analysis ............................................. 26
   3.6 Join-Less Approach based on Co-location Pattern Instance(CPI) Tree .................................................. 27
      3.6.1 Analysis ............................................. 30
   3.7 Join-Less Approach based on improved-CPI(i-CPI) Tree ........................................... 31
      3.7.1 Analysis ............................................. 34
   3.8 Mining co-location rules without prevalence threshold .................................................. 35
      3.8.1 Using Maximal Participation Index ............... 35
      3.8.2 Finding N-Most Prevalent Co-located Event Sets/Patterns .................................................. 37
   3.9 Comparative Analysis .................................... 42

4 Open Problems: ................................................ 46
# List of Figures

1. Colocation Patterns - Illustration ........................................... 5
2. Neighborhood Enumeration: (a) Reference Centric Model (b) Window Centric Model (c) Event Centric Model ............................ 7
3. Illustration of Join-based approach ......................................... 15
4. Illustration of Partial-Join approach ....................................... 19
5. Neighborhood Materialization .................................................... 23
6. Star Neighborhood Enumeration ............................................... 23
7. Illustration of Joinless Approach .............................................. 25
8. Illustration of Joinless Approach .............................................. 28
9. Illustration of i-CPI Tree Construction ...................................... 33
10. Candidate Generation in i-CPI Tree Approach .......................... 34
11. Co-location Instance Generation is i-CPI Tree Approach .......... 34
12. Neighborhood Transactions ..................................................... 39
13. Event Neighborhood Transactions .......................................... 40
14. Event Trees ........................................................................... 40
15. Star Candidates ....................................................................... 41
16. Clique Candidates .................................................................... 41
List of Tables

3.1 Sensitivity of various approaches to parameters . . . . . . . 45
Co-location rule mining refers to identification of a subset of features that are frequently located together from a given collection of boolean spatial features and generation of confident rules from such patterns. These rules predict the presence of a set of features based on the existence of another set of features which are disjoint from the first set. The co-location rule mining has applications across many domains such as location-sensitive recommendations, ecology and environment, health, public safety, business, transportation and tourism amongst others. This report defines the problem of co-location rule mining and describes why it is difficult to apply standard association rule mining techniques in the spatial context. The report discusses various approaches designed to extract interesting co-location rules from a collection of boolean spatial features in a spatial database. This report presents the complexity involved in these approaches and also compares all these approaches across several design parameters. Finally the report identifies the potential open problems in this area.
Chapter 1

Introduction

The advancements in technology have facilitated us with a lot of location related information. For example use of GPS gives the location of an object and use of satellites presents us with raster images of earth’s surface at different times using which heat maps of earth on features such as temperature and pressure can be generated. In this way a lot of location information is being generated every day. Other examples of location related information include census data of India, and weather data of India for the past few years. This huge data led to the use of spatial databases for the proper organization of generated information.

The increased use of spatial databases to store vast amount of such location related information has brought the necessity for mining of interesting and potentially useful spatial patterns from these databases. This has led to interesting research in the area of spatial data mining. However, the standard techniques of classical data mining can not be applied to spatial data mining because of the special properties that spatial data sets possess. Various methods were proposed and extensions to standard techniques of data mining and machine learning were established over past few years to incorporate these exclusive properties of spatial data. Classification, clustering, co-location mining, hotspot discovery and outlier detection are various areas that come under the broad topic of spatial data mining. All of these have wide-range applications in many areas.

Co-location rule mining is one such area of spatial data mining. The identification of co-location patterns refers to finding the subset of features that are located together frequently from a large collection of boolean spatial features. These patterns are then used to extract confident co-location rules using which one can predict the existence of some features given the existence of some other features which are different from the first set of features.

Co-location rule mining has got many application across domains like
location-sensitive advertisements, genetics, transportation, tourism, ecology, weather prediction, and crime prediction. For example, a mobile service provider who has data of service patterns used by geographically nearby users and can provide location sensitive offers which boosts his sales. Of late, genetic field of biology is using the co-location mining for predicting the structures of DNA. Other examples are prediction of rainfall in a particular area based on climatic conditions of that area and other neighboring areas of it and prediction of possible crime in an area based on a series of events occurring in that area over a period of time. In this way there have been wide spread applications identified that use co-location mining and potential new applications are coming up every year.

Various approaches were proposed over the years for efficient generation of co-location rules from a given collection of spatial data sets. All these approaches have the notions of support and confidence thresholds which are used to identify co-location patterns and generate co-location rules respectively. The main challenge in co-location rule mining is generation of co-location instances and each of these approaches propose different algorithms for the efficient generation of co-location instances. All these methods addressing the same problem differ in the way of materializing the neighborhood relationship among boolean spatial features and method to generate co-location instances. Furthermore some approaches were proposed to identify co-location patterns without necessity for support threshold which is applicable in real world cases where user may not have idea of what value forms a good support threshold.
Chapter 2

Background

2.1 Spatial Data Mining (SDM)

Spatial Data Mining refers to the process of identifying potentially useful and interesting patterns from huge amount location related information stored in spatial databases. A pattern can be a rule or a representative statistic of data. These patterns can be used in predictions such as weather and crime. With the increased use of spatial databases, and with generation of huge amount of location information through the use of GPS-enabled devices and other earth satellites, huge demand arose for spatial data mining. Many applications including social networks, location aware news feed, location-sensitive recommendations find spatial data mining necessary. The broad topic of spatial data mining covers areas:

1. Classification
2. Association Rule Mining
3. Clustering
4. Spatial Outlier Detection and Hotspot Analysis

Though these techniques seem to be natural extension of classical data mining, because of the unique properties of spatial data described in the next section, traditional data mining techniques are inapplicable for spatial data. Over time many extensions for traditional methods have been proposed and most of them were discussed in [1]. However, all mining techniques mentioned above do not guarantee discovery of all useful patterns. For example, association rule mining guarantees discovery of all useful patterns whereas clustering does not.
2.2 Unique Properties of Spatial Data

There are three unique properties of spatial data that make the traditional techniques of data mining inapplicable to spatial data mining. They are:

1. In spatial data mining, the properties at a location are dependent on adjacent locations. For example, the temperature at a location is dependent on the temperature of nearby locations. The temperature change will be gradual in space but not drastic. Thus the traditional assumption of independently generated samples does not hold good for spatial data. This property is commonly referred to as “Spatial Auto-Correlation”.

2. The spatial data is not only non-independent but also non-identical in its distribution. For example, the distribution of rainfall is not identical across the globe.

3. The underlying space is continuous in SDM unlike in many traditional applications where the space under consideration is discrete.

Because of these special properties, classical data mining techniques can not be directly used for spatial data mining techniques and many extensions and modifications have been proposed over the years to incorporate these unique features of spatial data in traditional methods.

2.3 Co-location Pattern and Rule Mining

Co-location pattern and rule mining comes under association rule mining area of spatial data mining. In traditional data mining, the association rule mining is used to identify potentially interesting rules that imply togetherness from large data sets. For example, extraction of “What goes with What” rules from the market-basket data transactions enable businesses to provide offers accordingly. The co-location mining, though seems to be a natural extension of traditional association rule mining to spatial context, needs to be tackled through different methods because of unique properties of spatial data discussed above and also because of the absence of the notion of transactions in spatial data which is elaborated in the next section.

A spatial co-location pattern is defined as subset of boolean spatial features whose instances are closely located frequently in geographical space. A boolean feature implies that the feature can only be present or absent but can not take any values. For an example in the real world, the Nile Crocodile and Egyptian Plover species co-exist frequently in nature. The following figure
1 shows the instances of four different boolean spatial features represented by four different symbols. A careful examination identifies two co-location patterns namely \{`+','x'\} and \{`o','*'\}. Prevalent co-location patterns are generated from huge data sets using support threshold.

Figure 1: Colocation Patterns - Illustration

Spatial co-location rules are models to associate the presence of a set of boolean spatial features based on the existence of another set of boolean spatial features which is disjoint from the first set. For example, the co-location rule ‘Nile Crocodile → Egyptian Plover’ indicates that whenever there is a Nile Crocodile, there is a high chance of having Egyptian Plover in the nearby region. Co-location patterns identified from large spatial databases are used to derive the co-location rules. Confident co-location rules are generated from prevalent co-location patterns using a confidence threshold.

Co-location mining is sensitive to the type of data (point, line or polygon) and neighborhood definition (distance metric or topological metric). Co-location pattern and rule mining have found applications across many domains as explained in the first chapter.
2.4 Difference between co-location mining and traditional association rule mining

The association rule mining is used to find interesting patterns from large databases such as market-basket transactions data. Identifying the items that go together in market transactions help businesses to design offers accordingly. Though co-location mining appears to be a natural extension of association rule mining in spatial context, the traditional techniques for association rule mining can not be directly applied to co-location mining. The reasons for this are:

1. The unique features of spatial data as discussed in above sections

2. There is no natural notion of transactions in spatial data which is inherent in market transactions. As the space under consideration is continuous, the transactions are not well-defined in spatial context.

Artificially transactionizing the space may lead to over or under estimation of interestingness measures such as support and confidence. And this makes the traditional techniques of association rule mining such as APRI-ORI inapplicable to co-location mining directly. Variants based on these techniques with extensions to incorporate the unique properties of spatial data have been proposed over the years. These techniques replace the notion of transactions with neighborhood and some of them are presented in the next chapter.

2.5 Neighborhood Enumeration

Various statistical methods such as correlation coefficients of instances were proposed for co-location rule mining but such methods are computationally expensive because of exponentially large number of instances of a given collection boolean spatial features. The notion of transactions is replaced by neighborhood in co-location mining. There are three basic ways by which the neighborhoods are identified that are discussed in [2]. They are as follows:

1. Reference Centric Model: Under this model a reference feature is identified and transactions are built around this feature. The remaining features are referred to as task-relevant features. For example in figure 2a, the reference feature is set to be A and B, C form task relevant features. An instance of B or C is said to be a neighbor of an instance of A if it is in the adjacent cell of A’s instance. In this example, for the
A’s instance at (2,3), B’s instances at (1,4) and (3,4) and C’s instances at (1,2) and (3,3) form neighbors. Thus for this instance of A, the transaction is \{B, C\}. This setting is useful in applications where there is focus on specific boolean spatial feature such as a disease and one needs to find what environment co-exists with that disease.

2. **Window Centric Model:** In this model, a window of fixed size is defined and this window is used to define transactions with or without boundary effect. The confidence of a pattern is defined as the number of windows in which the pattern occurs out of all possible windows. For example in figure 2b below, the space is divided into a number of overlapping windows of size 3x3. There are a total of 16 such windows and feature A’s instances are present in 15 of them whereas both A and C are present in 14 of them. So the rule of type ‘instance of feature A in a window → instance of feature C in window’ has confidence of $\frac{14}{15} = 0.93$ probability. This setting is useful for applications like surveying where presence of a feature is estimated given the presence of another feature in a land parcel and other applications involving administrative boundaries.

![Figure 2: Neighborhood Enumeration](image)

**Figure 2:** Neighborhood Enumeration: (a) Reference Centric Model (b) Window Centric Model (c) Event Centric Model

3. **Event Centric Model:** In this model the idea is to find the likelihood of occurrence of a subset of boolean spatial features around instance of given subset of features which are disjoint from first set by using the notions of instance neighborhood. The rules generated will have confidence based on how many instances of relevant feature types co-exist under the defined neighborhood notion. For example in figure 2c above, there are 4 instances of feature A and two of them have instances of C as neighbors assuming 9-neighbor neighborhood definition. Thus the rule ‘instance of feature A at a location → instance of feature C at the
same location’ has a confidence of 50%. The generation of transactions in this setting is sensitive to neighborhood definition. This setting has application in areas such as ecology where scientists want to find out the rules on co-existence of certain species.

2.6 Problem Statement

The following is the generic problem statement for co-location mining involving support threshold and this is the base for first five approaches discussed in the next chapter. For the last two approaches there is no necessity of specifying minimum prevalence threshold($min_{prev}$). And they do not generate all the co-location patterns and rules unlike the first five approaches.

**Input:**

1. A set of boolean spatial features $F = \{f_1, f_2, ..., f_k\}$
2. A set of instances $I = \{i_1, i_2, ..., i_n\}$ where each $i \in S$ is a vector $<$ instance id, featuretype, location $>$ where feature type $\in F$ and location belongs to spatial framework $S$
3. A neighborhood relation $R$ over the spatial framework $S$
4. A minimum prevalence threshold ($min_{prev}$) and a conditional probability threshold ($min_{cond\_prob}$) (These interestingness terms are defined in next section)

**Output:**

Find correct and complete set of co-location rules with participation index $> min_{prev}$ and conditional probability $> min_{cond\_prob}$

**Objective:**
Maximize computational efficiency

**Constraints:**

1. R is reflexive and symmetric
2. R is a distance metric based neighborhood relation
3. Monotonic prevalence measure
4. Spatial data set is a sparse point data set
2.7 Other Basic Concepts

The basic definitions and interestingness measures related to co-location mining which are discussed in [2] are explained below. These are common for all the approaches presented in next chapter.

2.7.1 Definitions

Definition 1: A co-location is a subset of boolean spatial features that exist together in space.

Definition 2: A co-location rule has the form: \( C_1 \rightarrow C_2(p, cp) \) where \( C_1 \) and \( C_2 \) are co-locations, \( p \) is a value representing the prevalence measure of the pattern and \( cp \) is a value representing conditional probability of the rule.

Definition 3: A neighborhood of a location \( l \) is a set of locations \( L = \{l_1, l_2, ..., l_k\} \) such that \( l_i \) is a neighbor of \( l \) i.e. \( R(l, l_i) \) is true \( \forall i \in 1, ..., k \).

Definition 4: If a subset of locations \( L' \) is a neighborhood of every location in a subset of locations \( L \), then \( L' \) is said to be neighborhood of \( L \).

Definition 5: \( I = \{i_1, ..., i_k\} \) is a row instance of a co-location \( C = \{f_1, ..., f_k\} \) if \( i_j \) if an instance of feature \( f_j (\forall j \in 1, ..., k) \) and \( I \) is a neighborhood of itself (meaning instances form a neighborhood clique).

For example in figure 2c, a row instance of co-location \{A,B\} is \{(2,3), (3,4)\}.

Definition 6: The table instance of a co-location \( C = \{f_1, ..., f_k\} \) is the collection of all its row instances.

2.7.2 Interestingness Measures

The interestingness measures for identification of co-location patterns and generation of co-location rules are defined as participation index and conditional probability respectively. The participation index is again defined in terms of participation ratio as minimum of the participation ratios of the features in a co-location pattern. The higher the participation index of a co-location pattern, the higher the chance of observing a subset of features of a pattern given the remaining subset of features in that pattern.

Formal definitions of interestingness measures are as follows:

Definition 7: The participation ratio \( pr(C, f_i) \) of a feature type \( f_i \) of a co-location \( C = \{f_1, ..., f_k\} \) is the fraction of instances of \( f_i \) which participate in any row instance of co-location \( C \). It can be formally defined as \( \frac{|\text{distinct}(\pi_{f_i}(\text{allrowinstancesof} C))|}{|\text{instancesof} f_i|} \) where \( \pi \) is a relational database projection operation.
Definition 8: The participation index of a co-location $C=\{f_1, ..., f_k\}$ is $\min\{pr(C, f_i)\}$.

The participation index exhibits anti-monotonicity property which implies that the value of participation index either decreases or remains the same as number of features in the co-location pattern increases. This is evident as the number of features increase by one, participation index of the pattern changes if and only if the participation ratios of any of the features is less than the current participation index by definition.

Definition 9: The conditional probability of a co-location rule $C_1 \rightarrow C_2$ is

$$\frac{|\text{distinct}(\pi_{C_1}(\text{allrowinstancesof}C_1 \cup C_2))|}{|\text{instancesof}\{C_1\}|}$$

For example consider two features A and B with 4 instances each. Let the co-location $\{A,B\}$ has row instances $\{(a.1,b.1),(a.1,b.2),(a.2,b.3)\}$. Then the participation ratio of A is 0.5 as only two instances of it participate in row instances of $\{A,B\}$ and participation ratio of B is 0.75. So the participation index of co-location pattern $\{A,B\}$ is $\min\{0.5,0.75\}=0.5$. The conditional probability of the rule $A \rightarrow B$ is 0.5.
Chapter 3

Various approaches for Co-location rule Mining and their analysis

All the approaches presented below address the co-location rule mining problem. The first five approaches deal with generating prevalent co-location patterns and confident co-location rules using prevalence and confidence thresholds. The last two approaches deal with generating confident co-location rules without any necessity for prevalence threshold. The steps of generating co-location candidates through apriori, pruning based on prevalence threshold once the instances are generated and generation of co-location rules using conditional probability threshold are same for the first five approaches (CPI-tree method differs in using prevalence measure). The main differences in all these approaches lie in the way they materialize the neighborhood relationships among instances (pre-processing) and the way they generate co-location instances of candidate co-locations. In the first approach all the steps are presented in a detailed manner and for the rest of approaches only those details where they differ from the first one are presented. In all the approaches discussed below, the terms features and events, instances and objects are used interchangeably.

3.1 Brute-force Approach:

The problem of generating co-location instances from given spatial data sets can be accomplished by generating all possible subsets of instances and checking if the elements of each of them form neighbors to one another. But there are many subsets that are unnecessarily enumerated such as \{A.1, A.2, B.1\}
which are not valid co-location candidate instances because of the repetition of a feature type. This method necessitates evaluation of neighborhood relationship checks for $2^n - n - 1$ subsets (excluding empty set and size-1 subsets), where $n$ is the number of instances.

In the worst case of evaluating all $2^n - n - 1$ subsets including those which are unnecessary, the computational complexity is exponential.

Number of neighborhood checks required =

$$\sum_{i=1}^{2^n-n-1} \frac{|i|(|i| - 1)}{2} = \frac{1}{2} \sum_{i=1}^{2^n-n-1} |i|^2 - \frac{1}{2} \sum_{i=1}^{2^n-n-1} |i| \rightarrow \Omega(2^n)$$

which is exponentially very high and is intractable as $n$ increases.

### 3.2 Posing as Graph Problem:

If the spatial instances can be represented as vertices of a graph with edges between two instances when they are neighbors, we get a graph representation of the problem. Here the problem of identifying co-location instances boils down to enumerating all cliques of the graph. But it has been proved to be a $NP-Hard$ problem. Also all identified cliques are not necessary as in above case. For example a clique representing $\{A.1, B.1, B.2\}$ is not necessary as it is not a valid candidate co-location instance.

### 3.3 Join-Based Approach

The join-based approach is discussed in [2] and [3]. This approach uses generalized-apriori (an extension of traditional apriori) to identify candidate co-locations and prune them using prevalence threshold based on the apriori property. Then the algorithm uses a hybrid method of geometric and combinatorial approaches to efficiently generate the co-location instances of different sizes. The candidate co-locations are pruned using participation index threshold and finally co-location rules are generated using conditional probability threshold. The algorithm employs the use of bitmap indices and generation of co-location rules in the increasing order of size by storing table instance size of lower order co-locations so as to efficiently perform participation index computation and confident rule generation. The detailed explanation of the algorithm with an example is given below.

1. Initially all size-1 co-locations are prevalent since all of them have a participation index of 1. So the first step initializes size-1 co-location instance set to the input $P$. 

12
For example in figure 3, for size-1 co-locations all features and their instances are retained as they are all prevalent.

2. Because all of the size-1 co-locations are prevalent, all possible size-2 combinations (in lexicographic order) form the candidate co-locations. The generation of size-2 co-location instances can be accomplished efficiently by geometrical approach such as plane sweep rather than combinatorial approach because all input instances form size-1 co-location instances without any pruning. All instances generated in such manner have to be sorted lexicographically as in case of candidate co-locations in order to use in the next step.

The candidate co-locations participation indices are computed and those with participation index < \textit{min_prev} will be pruned. Finally co-location rules of size-2 are generated from prevalent co-locations by filtering all possible size-2 rules against conditional probability threshold \textit{min_cond_prob}.

For example in figure 3, size-2 co-location instances are generated by geometric join of size-1 co-location instances and their participation indexes are computed. Based on the prevalence threshold, the candidate co-locations are pruned.

3. For co-location rules of size > 2, the algorithm loops so as to generate co-locations using \textit{generalized_apriori_gen}, generate co-location instances using combinatorial methods and then generate prevalent co-locations using which confident co-location rules are generated. The loop breaks when for any size, the prevalent co-location set is empty. Each of these steps are detailed below:

(a) Generation of candidate co-locations of size-\(k+1\) from prevalent co-locations of size-\(k\) is performed using \textit{generalized_apriori_gen} which is an adoption of traditional \textit{apriori_gen} algorithm. The \textit{generalized_apriori_gen} takes size-\(k\) prevalent co-location \(C_k\) and joins with another \(C_k\) as represented in following SQL-like statement.

\begin{verbatim}
insert into \(C_k+1\)
select \(p.feature_1, ..., p.feature_k, q.feature_k, p.table_instance_id, q.table_instance_id\)
from \(C_k p, C_k q\)
where \(p.feature_1 = q.feature_1, ..., p.feature_{k-1} = q.feature_{k-1}\),
\end{verbatim}
\[ p.feature_k < q.feature_k; \]

The last two columns are meant to keep track of table instances to use in generation of co-location instances in next step. Then using the apriori property algorithm prunes co-locations \( c \in C_{k+1} \) such that some subset of size-k of \( c \) is not in \( C_k \). This pruning step holds good because of anti-monotonicity property of the prevalence threshold. It can be shown as:

\[
\text{forall co-locations } c \in C_{k+1} \text{ do} \\
\text{forall size } k \text{ co-location } s \text{ of } c \text{ do} \\
\text{if } (s \notin C_k) \text{ then} \\
\text{delete } c \text{ from } C_{k+1};
\]

For example, in figure 3 below, \( \{A, B\} \) and \( \{A, C\} \) are joined to form a candidate co-location \( \{A, B, C\} \) and if \( \{B, C\} \) is not prevalent, then \( \{A, B, C\} \) is pruned by apriori-property.

(b) Then all co-location instances of candidate co-locations are generated using the identified table instances through combinatorial methods. The combinatorial method is explained as query below wherein equality check is done for all \( (k-1) \) instances and neighborhood check is done for last instance. Those co-location instances that pass the neighborhood check on last instance will be retained in the table instances of size-\( (k + 1) \).

\[
\text{forall co-location } c \in C_{k+1} \\
\text{insert into } T_c \\
\text{select } p.instance_1, ..., p.instance_k, q.instance_k \\
\text{from } c.id_1 p, c.id_2 q \\
\text{where } p.instance_1 = q.instance_1, ..., p.instance_{k-1} = q.instance_{k-1}, \\
(p.instance_k, q.instance_k) \in R; \\
\text{end;}
\]

Then all co-locations of size-\( k + 1 \) with empty table instances will be eliminated from \( C_{k+1} \).

For example in figure 3 above, since \( \{A, B\} \) and \( \{A, C\} \) are joined to form co-location \( \{A, B, C\} \), the corresponding table instances \( t4 \) and \( t5 \) are joined in accordance with above notation to form table instance of \( \{A, B, C\} \). Here \( A.2, B.2 \) and \( A.2, C.1 \) join to form a co-location instance \( A.2, B.2, C.1 \) since \( (B.2, C.1) \) are neighbors.
Figure 3: Illustration of Join-based approach
(c) Then participation indices of the candidate co-locations retained are computed. To compute the participation index of a candidate co-location, by definition, we need to compute the participation ratios of each of the features in candidate co-location. To efficiently compute participation ratios, bitmaps are used. For each feature, a bitmap of length = no. of instances of feature is created with all 0’s. Then the table instance of corresponding candidate co-location is scanned once and for all the instances of that feature found in the table instance, the corresponding bits in bitmap are set to 1. Thus the participation ratio of the feature can be computed by dividing the number of 1’s in the bitmap with the size of bitmap of that feature. Once the participation ratios of all features are computed, the participation index of whole co-location candidate is minimum of the participation ratios of its features.

For example, the participation ratio of $A$ in $\{A, C\}$ is computed from bitmap 0111 (generated as explained above) as $0.75$ and the participation ratio of $C$ is computed from bitmap 111 as 1 and thus giving the participation index of $\{A, B\}$ as $\min\{0.75, 1\} = 0.75$.

Then all candidate co-locations whose participation index is less than prevalence threshold ($\text{min\_prev}$) are pruned. For the prevalent co-locations that pass this step, the table instances are stored for use in next iteration and discarded later. The sizes of table instances of prevalent co-locations are also stored for efficient computation of conditional probability of candidate co-location rules.

(d) For each prevalent co-location $C$, all possible rules of the form $C' \rightarrow C - C'$ are generated where $C'$ is a proper subset of $C$. For a rule $C' \rightarrow C - C'$, divide the cardinality of projected table instance of $C$ on $C'$ after duplicate elimination with cardinality of $C'$ (which is stored in previous iterations) to compute conditional probability and retain it as valid rule if that value exceeds the minimum conditional probability threshold ($\text{min\_cod\_prob}$).

The algorithm in this approach uses a join-based method to generate co-location rules. The algorithm is space-efficient in a way that it does not retain the intermediate results that are unnecessary. Bitmap techniques are efficiently employed to compute participation ratio by a single pass through table instance. Similarly, the cardinality of each prevalent co-location’s table instance is stored and co-location rules are generated in increasing order of size for efficient computation of conditional probability.
The algorithm necessitates neighborhood check for all instances in the table instance generation step which is computationally very intensive and that determines the overall performance of algorithm. Thus the algorithm is not scalable. Many approaches are proposed to address this expensive step and some of them are discussed in the next sections of this chapter.

### 3.3.1 Analysis

The most expensive step of join-based algorithm is co-location instance generation. While joining two table instances of size-$k$ to generate a size-$(k + 1)$ table instance, a pair of co-location instances from two table instances are checked if the first $(k - 1)$ instances match and if the last instances in both co-location instances form neighbors. The worst case occurs when both table instances have all their co-location instances with identical first $(k - 1)$ instances. In this case each co-location instance of a table instance must be checked for neighborhood relationship with every co-location instance of second table instance. So, in the worst case if there are $m$ and $n$ co-location instances in size-$(k)$ table instances to be joined, we need $O(mn)$ neighborhood checks while joining two table instances. And the total complexity increases based on the number of instances and length of co-location patterns in the data set.

### 3.4 Partial-Join Approach

In order to reduce the number of joins required to generate co-location instances in the join-based algorithm, which forms the bottleneck of it and makes it non-scalable, a partial-join approach is proposed in [4]. In this approach, the neighborhood is enumerated in the form of clique instances(divided into cliques and each clique is identified as a transaction) and then only those instances whose relationship cuts across cliques need to be joined. For all other instances generation of table instances of any level is simple and straightforward using transaction based \textit{apriori} algorithm. This way the number of joins required will be reduced and the performance improvement is significant when the number of neighborhood relations across cliques are too less. The algorithm is detailed below along with an example and definitions needed.

**Definition 1:** A \textit{neighborhood transaction} (or \textit{transaction}) is a subset of instances that forms a clique using neighborhood relation $R$. The set of instances $S$ is divided into mutually exclusive and exhaustive subsets $T_i \subset$
For example consider figure 4 where the same set of instances shown in first approach are divided into transactions. The transactions are depicted by cliques within dashed circles. In figure 4, \{A.2, B.2, C.1\}, \{A.3, B.3, C.2\}, \{A.1, B.1\}, and \{A.4, C.3\} form transactions. A naive way to transactionize the space is to partition it into a grid of size $d \times d$ where $d$ is threshold distance to quantify neighbors.

**Definition 2:** A row instance $I$ of a co-location $C$ is said to be an **intra-X row instance** (or **intra-X instance**) if all instances $i \in I$ belong to a single transaction $T$. The collection of all such intra – X row instances of a co-location forms **intra-X table instance**.

For example in figure 4, for the co-location \{A, B\}, the row instance \{A.1, B.1\} forms intra – X instance and the intra – X table instance of \{A, B\} is a set consisting of \{A.1, B.1\}, \{A.2, B.2\} and \{A.3, B.3\}.

**Definition 3:** A neighbor relation $r \in R$ between two instances $i_1$ and $i_2$ is said to be a **cut-neighbor relation** (or **cut-relation**) if $i_1$ and $i_2$ are neighbors of each other but exist in different transactions.

For example in figure 4, the cut-relations are shown in dotted lines. In this example, the neighborhood relation between A.2 and B.1 is a cut-relation. cut-relations in a grid based transactionization can be enumerated by checking neighbor instance pairs that span across cells.

**Definition 4:** A row instance $I$ of a co-location $C$ is said to be an **inter-X row instance** (or **inter-X instance**) if all instances $i \in I$ have at least one cut-neighbor relation. The collection of all such inter-X row instances of a co-location forms **inter-X table instance**.

For example in figure 4, \{A.2, B.1\} form an inter-X instance of \{A, B\} and its inter-X table instance is a set consisting of \{A.2, B.1\} and \{A.4, B.3\}. In this example the co-location instance \{A.2, C.1\} forms both intra-X instance and inter-X instance.
Figure 4: Illustration of Partial-Join approach
The algorithm in this approach works as follows:

1. The algorithm first transactionizes the space using some heuristic such as grid-based transactionization. The ideal way of transactionizing instances would be to have maximal cliques in transactions with minimal number of neighborhood relations across clique. In 4, the transactions are made by drawing circles with arbitrary centers and certain radius.

2. Then the candidate co-locations are generated using generalized \textit{apriori} where feature level pruning is done using anti-monotonicity property of participation index.

3. The transactions are then scanned and all \textit{intra-X instances} of a candidate co-location are generated. There is no need for any join or neighborhood check because all instances of a transaction form a clique. This step can be efficiently done by employing some bitmap techniques in combination with trie data structure.

   For example in figure 4, an intra-X instance of \{A, B\} is \{A.1, B.1\}.

4. Then all \textit{inter-X instances} of size-(\(k + 1\)) (where \(k \geq 2\)) candidate co-locations are generated using the the \textit{inter-X instances} of size-\(k\) in a way similar to join-based approach where the combinatorial methods are used. That is, equality check on first \(k - 1\) instances and neighborhood check on \(k^{th}\) instance is performed. Those instances that pass the filter are retained in the \textit{inter-X table instance} of candidate co-location.

   For example in figure 4, the \textit{inter-X instance} of \{A, B, C\} is generated by joining \textit{inter-X instances} \{A.2, B.1\} and \{A.2, C.1\} as \{A.2, B.1, C.1\} since A.2 and B.1 are neighbors.

5. Then the participation index of a candidate co-location is computed by combining both \textit{intra-X instance} and \textit{inter-X instance}. The bitmap techniques used in join-based approach can be extended to this approach for efficient computation of participation index. For example, in figure 4, for the candidate co-location \{B, C\}, the participation index is \(\min\{1, 0.66\} = 0.66\). The candidate co-locations whose participation index is less than the prevalent threshold are pruned at this stage.

6. Finally all possible size-\(k\) co-location rules are evaluated against the conditional probability threshold(\(\min_{\text{cond. prob}}\)) in order to generate confident co-location rules using techniques similar to join-based approach presented in the previous section.
This approach essentially addresses the bottleneck of previous join-based approach by cutting down on the number of joins to be performed in order to generate co-location instances. It does so by transactionizing the space and then using apriori property effectively to separately generate *intra-X instances* and *inter-X instances*. And the savings are significant when the number of cut-relations is very less. The approach also employs bitmap techniques for computation of participation indices. Also the efficient method of generating co-location rules in the last stage is retained here.

However, the way of ideal transactionization of space is not clear from the algorithm. Also there are certain instances which fall under both *intra-X instances* and *inter-X instances* which leads to double enumeration of co-location instances of certain size (which does not occur in join-based approach). Moreover, this approach is as bad as join-based approach when the number of cut-relations is very high. The worst case occurs when each instance forms as separate transaction in which case the performance of this approach will be same as that of join-based approach.

### 3.4.1 Analysis

The most expensive step of this approach is generating co-location instances. And the complexity depends entirely on the number of cut-relations which is decided by the method using which we transactionize the space. In the worst case each instance in transactionized into a single transaction and all relations will be cut-relations. In such case this approach is equivalent to join-based approach. Then while joining two size-\((k - 1)\) table instances of sizes \(m\) and \(n\) to generate a table instance of size-\(k\), the number of neighborhood checks needed will be \(O(mn)\) as in the case of join-based approach. Similar to that case, the complexity increases based on the number of instances and length of co-location patterns in the input data set.

If the cost of generating clique neighborhoods from the data set during pre-processing is too high, then it may make the overall partial-join approach more costlier than join-based approach in the worst case described above. It is so because of the added cost of clique neighborhood enumeration in the worst case and extra processing needed for intra-X instances. In such a case join-based approach which does not require any pre-processing performs better.
3.5 Join-Less Approach

This approach presented in [5] and [6] tries to completely eliminate the necessity of joins to produce co-location instances by using instance lookup scheme. And so it overcomes the bottlenecks in join-based and partial-join approaches of co-location rule mining. It materializes the neighborhood relationships through star neighborhood partition model without duplication and loss of co-location instances. Star instances of each candidate co-location are generated which form the superset of co-location instances. Then co-location instances are generated from star instances by using star neighborhoods without any necessity of joins. The final steps of prevalent co-location identification and co-location rule generation are same as in previous methods. The approach also uses coarse level filtering on star instances to prune candidate co-locations without actually generating co-location instances. The algorithm is explained with an example below along with necessary definitions.

**Definition 1:** For a spatial instance \( i \in S \) of feature type \( f_i \in F \), the **star neighborhood** of \( i \) is a set of spatial objects \( T = \{ i_j \in S | i_j = (i_j \lor (f_i < f_j \land R(i_i, i_j))) \} \) where \( f_j \in F \) is the feature type of \( i_j \) and \( R \) is a neighbor relationship.

In other words, the star neighborhood of an instance is a set of center instance and other instances in its neighborhood whose feature types are greater than the feature type of center instance in the lexical order.

For example in figure 5 below, the neighborhood areas of instances A.1, B.2 and A.4 are shown where the dotted circles have radius \( d \) (distance threshold) with center as particular instance. In this example for A.4 the neighborhood area contains B.3 and C.3. And so the star neighborhood of A.4 is \{A.4, B.3, C.3\}. Similarly for B.2, the neighborhood area contains A.2 and C.1. But since A.2 is lexicographically lower than B.2, it is not counted in its star neighborhood and so the star neighborhood of B.2 is \{B.2, C.1\}. The detailed star neighborhoods of all instances are shown in figure 6 below.

**Definition 2:** Let \( I = \{i_1, ..., i_k\} \) be a set of spatial instances whose feature types respectively are \( \{f_1, ..., f_k\} \). If all instances in I are neighbors to the first instance \( i_1 \), I is called a **star instance** of co-location \( C = \{f_1, f_2, .., f_k\} \).

From figures 5 and 6, we can see that a subset of star neighborhood of A.3 namely \{A.3, B.3, C.2\} forms the star instance of \{A, B, C\}.
The algorithm in this approach works as follows:

1. The algorithm first enumerates the star neighborhoods of each of the instances from the given input data set using neighborhood relationship criterion.

2. The candidate co-locations of size-1 and size-2 are generated directly whereas those of size $> 2$ are generated using \textit{generalized \text{\_} apriori} where feature level pruning is done using anti-monotonicity property of participation index. This step is similar to previous approaches.

3. For each candidate co-location, the star instances are generated from star neighborhoods whose center instance feature type is same as the
first feature of the candidate co-location.

For example from figure 6, we can enumerate a star instance of \( \{A, B, C\} \) as \( \{A.4, B.3, C.3\} \). The detailed enumeration of all star instances and further processing is shown in figure 7.

4. Once the star instances of a candidate co-location are enumerated, the algorithm performs coarse-level pruning of candidates based on the fact that the participation index of the star instances of any co-location is greater than or equal to that of its co-location instances. This holds good because the star instances form the superset of co-location instances.

For example, figure 7, shows that the participation index of candidate co-location \( \{A, B, C\} \) is 0.75. If the prevalence threshold is greater than this value, then the candidate co-location \( \{A, B, C\} \) is pruned at this stage without actually enumerating its co-location instances.

5. If the candidate co-location passes the filter step at star instance level, then actual co-location instances are enumerated. To do so, we have to check if all of the instances in the given star instance form a clique. For first instance, all others are neighbors by definition of star instance. For the rest of instances we need to check if each instance satisfies the neighborhood relationship with higher order instances in that particular star instance using star neighborhoods. If all of them do, then the star instance is a co-location instance and if at least one instance does not satisfy neighborhood check then it can not be treated as co-location instance.

For example, in figure 7, for the first and second star instances of candidate co-location \( \{A, B, C\} \), B.1’s star neighborhood has C.1 but not C.2. Hence \( \{A.1, B.1, C.1\} \) forms co-location instance of \( \{A, B, C\} \), but \( \{A.1, B.1, C.2\} \) does not.

6. Once the co-location instances of candidate co-locations are enumerated, the participation indexes are computed to select the prevalent co-locations of size-k and then generate the confident co-location rules of size-k from them. The techniques for participation index computation, rule generation are same as that of in join-based and partial-join approaches.

This approach addresses the problem with previous approaches which necessitate some join computations to identify co-location instances. The joinless algorithm does away with join computations by using instance lookup
Figure 7: Illustration of Joinless Approach
scheme for generating co-location instances. This brings in significant savings in computation time. Moreover, the algorithm performs coarse-level filtering eliminating the necessity for generation of unnecessary co-location instances. The bitmap index techniques for efficient computation of participation index and confident co-location rule generation can be extended to this approach.

However the generation of co-location instances from star instances necessitates further lookups and the total cost becomes significant when the data set has long co-location patterns as well as large number of instances.

3.5.1 Analysis

The generation of co-location instances from star instances is the expensive step in this approach and is significant especially when the data has large number of instances and long co-location patterns. In the worst case, the filter at star instance level fails and all star instances of all sizes need to be check for probable co-location instances and all happen to be co-location instances. In such a case, for one size-$i$ star instance to be ratified as co-location stances the number of lookups needed are:

$$[(i-2) + (i-3) + ... + 1] = \frac{(i-2)(i-1)}{2} \times t_{\text{lookup-time}} \rightarrow O(i^2)$$

For all size-$i$ star instances the number of lookups needed are:

$$|i| \times \frac{(i-2)(i-1)}{2} \times t_{\text{lookup-time}}$$

where $|i|$ denotes number of star instances of size-$i$. The final cost of whole algorithm includes star neighborhood generation cost and generation of co-location instances of all sizes. Then the worst case complexity of overall algorithm is (considering only dominant steps):

$$\sum_{k=1}^{k_{\text{max}}} O(|\text{starneighborhood}_k|) + \sum_{k=1}^{k_{\text{max}}} |C_k| \times \frac{(k-2)(k-1)}{2} \times t_{\text{lookup-time}}$$

where $O(|\text{starneighborhood}_k|)$ denotes the time to generate star neighborhoods of size-$k$, $|C_k|$ denotes to total number of instances of all co-locations of size-$k$ and $k_{\text{max}}$ is the maximum length of co-location patterns in the given data set.

The star neighborhood partition in this approach partitions the edges dis-jointly whereas the clique neighborhood partition in partial-join approach partitions the nodes dis-jointly. There will be $m$ star-neighborhoods - one for
each instance whereas the number of clique partitions depends on the partitioning method chosen. In the worst case there will be \( m \) clique partitions when each instance is identified in a different transaction. But in general the number star neighborhoods is larger than the number of clique partitions. Enumeration of star-neighborhoods is easy but they do not directly model cliques which imply co-location instances. But the clique partitions directly model the cliques but are difficult to enumerate. If the cost of generation of star neighborhoods from the data set during pre-processing is very large, it may sometimes compensate the benefits of join-less lookup based approach.

### 3.6 Join-Less Approach based on Co-location Pattern Instance (CPI) Tree

This is another join-less approach to generate co-location instances without any necessity for expensive join computations and is presented in [7]. It addresses the problem in previous join-less approach where the computation time of generating table instances increase significantly with the length of co-location patterns and the number of co-location instances because of necessity to filter co-location instances from star instances. The algorithm uses the graph \( G \) constructed using the neighborhood relationship criterion on instances as shown in 3. The approach uses the Co-location Pattern Instance (CPI) tree to materialize neighborhood in a compact way. The algorithm then uses the recursive and hierarchical properties of tree data structure to enumerate co-location instances. The algorithm directly enumerates size-\((k + 1)\) co-location instance from corresponding size-\(k\) co-location instances without waiting till all size-\(k\) co-location instances are enumerated. The detailed algorithm with example and required definitions is described below.

**Definition 1:** A **CPI-tree** is a rooted tree whose root is labelled as “null”. A branch of CPI-tree corresponds to a connected sub-graph in graph \( G \). Each node in the tree represents a co-location instance. Node \( u \) is the parent of node \( v \) in the tree, if \( u \) and \( v \) form neighbors in graph \( G \) and \( u \) is lexicographically smaller than \( v \). An indirect(dashed) link is made in the tree when a co-location instance is repeated from the repeated occurrence to the first occurrence so as to avoid duplication of relationship materialization.

The algorithm first generates a sorted set of neighbor instances of each instance such that the set contains only those neighbor instances that are lexicographically greater than the instance under consideration. Using these
sets and stack data structure the algorithm builds a tree that satisfies the afore mentioned properties of CPI-tree. The algorithm builds the tree in depth-first manner; that is each branch of the root is generated from corresponding connected sub-graph in $G$ and a branch is constructed only after the previous branch is completely constructed. The CPI-tree constructed in such a manner has no duplication and no loss of co-location instances and is unique. The CPI-tree generated for the graph in 3 is shown in figure 8. There is an indirect link from the second occurrence of instance $B.1$ to its first occurrence.

**Definition 2:** A link between two nodes in a CPI-tree is called as **direct-child link**. A direct-child link of length $k$ is called **size-$k$ direct-child link**. Those nodes lying in the middle of size-$k$ direct-child link are called **intra-nodes**. For a given node in a direct-child link, the nodes lying along the link and below this node are referred to as **child-nodes** of the node.

For example in figure 8, $A.2 - B.2$ is a direct-child link, $A.3 - B.2 - C.1$ is a size-3 direct-child link, $B.2$ is intra-node of the size-3 direct child link $A.3 - B.2 - C.1$ and $B.3$ and $C.1$ are child nodes of $A.3$.

**Definition 3:** The child node through indirect(dashed in figure) link is called as **indirect-child**. A child link of length $k$ linked through an indirect-child is called **size-$k$ indirect-child link**.

For example, in figure 8, $C.1$ is the indirect-child of second occurrence of $B.1$ and they are joined by an indirect link shown as dashed link in the figure. Any size-$k$ child-link involving an indirect-child such as $C.1$ in this case forms a size-$k$ indirect-child link.

**Definition 4:** A size-$k$ direct-child link is called a **size-$k$ all-link** if all the child-nodes of each intra-node in a size-$k$ direct-child link are also brothers of that node. Similarly, a size-$k$ indirect-child link is called a **size-$k$ indirect-
**all-link** if all the child-nodes of each intra-node in a size-k indirect-child link are also brothers of that node.

For example in figure 8, the direct-child link $A.2 - B.2 - C.1$ is a size-3 all-link since $C.1$ is also a brother of intra-node $B.1$. Similarly, the indirect-child link $A.4 - B.3 - C.3$ is an indirect-all-link.

There are two properties that govern the CPI-tree and can be exploited to efficiently generate co-location instances from it. They are:

**Child-link property:** Each size-2 child-link (direct or indirect) in CPI-tree forms a size-2 co-location instance. This property holds good by construction.

**All-link property:** Each size-k all-link or size-k indirect-all-link in CPI-tree forms a size-k co-location instance. This holds good because when each intra-node’s child in a size-k direct (or indirect) child link is its brother, then we get a clique from that link.

For example, in figure 8, consider the size-3 direct-child link $A.2 - B.2 - C.1$. Since $C.1$ forms brother as well as child of $B.2$, $C.1$ will be a child of $A.2$. So there is a connection between every pair of instances in this link. Hence $\{A.2, B.2, C.1\}$ forms a co-location instance of the candidate co-location $\{A, B, C\}$.

Using these properties and above constructed CPI-tree, all table instances of given data set can be generated such that there is no duplication and loss of correct co-location instances. For any size-k co-location instance to be valid, all instances in it must form a clique which can be verified from the aforementioned two properties of CPI-tree. The algorithm generates co-location instances and then enumerates prevalent co-locations and confident rules in the following way:

1. Once the CPI-tree is built, the proposed approach generates the co-location instances recursively. It does depth-first search of each branch of root node to enumerate all possible co-location instances of any size.
   The algorithm first traverses recursively to the bottom of a size-k link in branch to enumerate size-2 co-location instance. It then travels upward along the size-k link enumerating co-location instances of all sizes that are valid using above properties.
   For example, consider the size-3 direct-child link $A.2 - B.2 - C.1$. The algorithm first generates size-2 co-location instance $\{B.2, C.1\}$ after recursively traversing to the bottom of the tree. It then moves back to previous level and generates co-location instance $\{A.2, B.2\}$. It then
checks the all-link property on $A.2 - B.2 - C.1$ and finds that $C.1$ is brother as well as child of $B.2$. So it enumerates $\{A.2, B.2, C.1\}$ as size-3 co-location instance. Similarly it enumerates $\{B.3, C.2\}$, $\{A.4, B.3\}$ as size-2 co-location instances and $\{A.4, B.3, C.2\}$ as size-3 co-location instance from the indirect-child link $A.4 - B.3 - C.2$.

The recursion process is extended till the first level node is reached thereby enumerating all possible co-location instances of all sizes from that link. In essence the algorithm is a backtracking process.

2. Once the co-location instances are identified, the generation of prevalent co-locations (though there is no prevalence level pruning of candidates, prevalent co-locations can be enumerated) and confident co-location rules is same as in all previous approaches. All the optimization techniques such as bitmaps used in previous approaches can be used here too.

This approach gives a less expensive way of identifying the co-location candidates using the CPI-tree data structure since it eliminates the necessity of filtering co-location instances from star instances as in previous approach. All co-location instances are efficiently and directly generated from CPI-tree using its properties and there is no need for any filtering.

But the approach unlike all previous approaches will not make use of apriori-property which generates size-$k$ candidate co-locations from size-$(k - 1)$ prevalent co-locations by pruning non-prevalent candidates thereby considerably saving the computation time. As this approach generates co-location instances of size-$k$ without generating all size-$(k - 1)$ instances, the apriori property can not be used. Thus it ends up generating unnecessary co-location instances. And this forms the bottleneck of the algorithm as the size of co-location patterns and the number of instances grow in the data set. Besides the approach is also inefficient because of indefinite number of recursions involved in generating co-location instances.

### 3.6.1 Analysis

Construction of CPI-tree has a complexity of $O(m^2 \log m)$ where $m$ is the number of instances. This is because the tree construction requires sorted neighborhood set generation for each instance which has the set of all instances that are neighbors to and lexicographically bigger than the instance under consideration. In the worst case, all instances are of different feature types and all of them form a clique. Then in the neighborhood set $i$, there
are \((m - i)\) elements. To sort them, the complexity is:

\[
\sum_{i=1}^{m} (m - i) \log(m - i) \rightarrow O(m^2 \log m)
\]

The rest of tree construction process takes \(O(m^2)\) because each instance is processed twice.

Since each neighborhood relationship is stored once in the tree, the space complexity is \(O(nbr relationships)\). So in the worst case when all instances form a clique, the space complexity will be \(O(m^2)\).

Enumeration of co-location instances from CPI-tree requires indefinite number of recursions. The enumeration happens in depth-first manner. In the worst case when all the instances of different feature types form cliques separately, we get a single branch for each such clique in CPI-tree and enumeration of all size co-location instances in that branch requires \(O(n^2)\) traversals down the branch where \(n\) is the number of instances that are forming a single clique. The total number of traversals is based on number of such cliques. If there are \(m\) such cliques, then the number of traversals is \(O(mn^2)\).

### 3.7 Join-Less Approach based on improved-CPI(i-CPI) Tree

This approach presented in [8] attempts to combine the merits of first join-less approach and CPI-tree approach. Essentially it provides an extension to CPI-tree approach by making some modifications to incorporate the apriori-property usage. And so it is also a join-less approach to generate co-location instances. It materializes the neighborhood relationships in an improved-CPI(i-CPI) tree in a way to be able to use apriori property. Hence a co-location candidate will be generated only if all its subsets are prevalent co-locations. This saves significant computation time without any need to generate unnecessary co-location instances especially when the size of co-location patterns in data set is very high. The algorithm then uses the recursive and hierarchical properties of i-CPI tree to generate co-location instances in increasing order of size unlike CPI-tree approach. The detailed explanation of algorithm along with examples and necessary definitions is given below.

**Definition 1:** Given a subset of spatial instance \(I = \{i_l, ..., i_v\}, l, v \in \{1, 2, ..., m\}\), if \(i_l \leq i_j\) (lexicographically) holds for all \(l \leq i \leq j \leq v\), then \(I\) is called as ordered instance set. If \(I\) is a valid table instance, then it is called ordered table instance. If the feature type of \(i_i\) is not same
as feature type of $i_l$, and $R(i_l, i_i)$ holds for all $l \leq i \leq v$, then $I$ is called **ordered neighbor relationship set** of the instance $i_l$. The set of ordered neighbor relationship sets of all instances of a spatial feature $x$ is denoted by $\delta_x$.

For example in figure 3, $I = \{A.2, B.2, C.1\}$ is an ordered instance set and also an ordered table instance. The ordered neighbor relationship set of instance A.2 is $\{A.2, B.1, B.2, C.1, C.2\}$. For feature A, the set of all ordered neighbor relationship sets of all instances is:

$$\delta_A = \{\{A.1, B.1\}, \{A.2, B.1, B.2, C.1, C.2\}, \{A.3, B.3, C.2\}, \{A.4, B.3, C.3\}\}$$

**Definition 2:** Given a set of spatial feature types $F = \{f_1, ..., f_n\}$ and a set of ordered instance neighbor relationship of features $\delta = \delta_{f1} \cup ... \cup \delta_{fn}$, $\delta_{f_i}(1 < i < n)$ is the set of ordered neighbor relationship sets of all instances of feature $f_i$, the **i-CPI tree** is a rooted tree whose root is labelled as “null” and has sub-trees for all but one(last) feature types. The sub-tree for spatial feature $f_i$ consists of the root $f_i$ and each subset of $\delta_{f_i}$ as branch of root. Each such branch records an ordered neighbor relationship set of corresponding instance and feature type.

The algorithm works as follows:

1. The algorithm first generates all spatial ordered neighbor relationship set for all features.

2. Using this set the i-CPI tree is constructed in depth-first manner building each sub-tree.

   For example for the data set in figure 3, we have the i-CPI constructed as in figure 9. The root of the tree is labelled “null”. It has a sub-tree for each feature type. Each sub-tree has a root named after the feature type and a branch for each instance of the feature type. Here there is a sub-tree for feature type A with branches for A.1, A.2, A.3 and A.4. A branch for A.2 captures the ordered neighbor relationship set of it as can be seen from the figure. Similarly using all other neighbor relationship sets except $C$ (as it is the last feature and so does not have any such set), the i-CPI tree is constructed as in figure 9.

   The i-CPI tree constructed for a given spatial data set will be unique. It will have no duplications since each neighborhood relationship is stored exactly once and there is no loss of co-location instances.

3. Once the i-CPI tree is constructed for a given data set, the algorithm iteratively generates candidate co-locations, generates table instances for them, prunes them and finally generates the confident co-location
rules of each size in increasing order of sizes. Size-$k$ candidate co-locations are generated from size-$(k - 1)$ prevalent co-locations using the apriori property as in normal join-less approach. This is shown in figure 10.

4. The table instances of each candidate co-location are enumerated from the i-CPI tree. Size-$k$ table instances are expanded from size-$(k - 1)$ table instances whose feature types are same as that of first $(k - 1)$ features of candidate co-location. The procedure is shown in figure 11.

For example, the co-location instances of candidate co-location $\{A, B, C\}$ from the co-location instances of $\{A, B\}$. The first step takes count of each of the instances of last feature in candidate co-location satisfying neighborhood relationship with the instance of $\{A, B\}$ under consideration. In this case the instance is $\{A.2, B.2\}$. Then the second step identifies the expanded list with those instances of the last feature of size-$k$ candidate co-location whose count in previous step is $k - 1$. Here C.1 has count 2 and so it is included in expanded list. Finally the table instances are formed by putting together the instance under consideration of size-$(k - 1)$ and each of the instances of expanded list. In our example, the co-location instance $\{A.2, B.2, C.1\}$ is identified by putting together $\{A.2, B.2\}$ and C.1.

5. After the instance generation, candidate co-locations can be pruned based on participation index and confident co-location rules can be generated in a way similar to all our previous approaches. All optimizations such as bitmaps usage are applicable in this approach also.
This approach addresses the problem of generating co-location instances in a join-less way and overcoming the bottleneck in CPI-tree approach. By preserving apriori property, candidate co-locations can be pruned at much coarse level thereby reducing necessary co-location instance enumerations. This brings significant savings when the data set has long co-location patterns. The approach also preserves the idea of CPI-tree approach to materialize neighborhood relationships in a tree and uses its hierarchical properties to generate co-location instances.

3.7.1 Analysis

The construction of i-CPI tree has time complexity of $O(m^2 \log m)$ for the same reasons discussed in case of CPI-tree where the sorted neighborhood sets of each instance are generated from the input data. Similarly the space complexity of i-CPI tree is $O(nbr - relationships)$ as each neighbor relation-
ship is stored once in the i-CPI tree. In the worst case when all instances form a clique, the space complexity would be $O(m^2)$ where $m$ is number of instances. For the generation of a co-location instance of size-$k$ from size-$(k - 1)$ co-location instance, the tree needs to be traversed once for each of the $(k - 1)$ instances. In the worst case when all instances of different feature types form cliques separately, to generate a co-location instance of size-$k$, we need to have $O(k)$ traversals below the feature type nodes. Overall, from size-1 to size-$k$ co-location instance generation of a single clique, we need traversals of $O(k^2)$. The overall cost depends on number of such cliques and if there are $m$ such cliques, then to generate all co-location instances of size-$k$ beginning from size-1, we need $O(mk^2)$ traversals below the feature type nodes.

### 3.8 Mining co-location rules without prevalence threshold

All above discussed approaches need the minimum prevalence threshold to be specified by the users. In some cases these methods do not generate all confident co-location rules because the participation index by definition favors the frequent features and so most likely prunes candidate co-locations with rare features at early stages. Similarly users may not be aware of what forms a good participation index in all cases. To address these two different scenarios, two methods are proposed which do away with necessity of minimum prevalence threshold. These two approaches are discussed in the following sub-sections.

#### 3.8.1 Using Maximal Participation Index

In the previously discussed approaches a set of feature types form a co-location if for each of the features, at least $s\%$ of instances form neighbors to instances of other features in the set. But this method has bias towards frequent spatial features compared to rare spatial features. The approach presented in [9] proposes and uses maximal participation index so as to generate confident co-location rules even if they do not form prevalent(popular) co-locations. The approach exploits the weak – monotonicity property of maximal participation index to generate confident rules.

**Motivating Example:** Consider a co-location pattern: $C = \{\text{cadmium contaminated water, respiratory problems, renal problems}\}$.  

35
Suppose \( pr(C, \text{cadmium contaminated water}) = 80\% \), \( pr(C, \text{respiratory problems}) = 2\% \) and \( pr(C, \text{renal problems}) = 1\% \). Then participation index \( PI(C) = \min\{80\%, 2\%, 1\%\} = 1\% \).

This co-location will most likely be pruned because of low participation index. However, \textit{cadmium contaminated water} has high implications to respiratory problems and renal problems making them confident rules. These rules will not be generated by the approaches discussed in the previous sections.

**Definition 1:** The \textit{maximal participation index} (\( maxPI \)) is defined as the maximum of participation ratios of a co-location.

\[
maxPI = \max_{f \in C} \{pr(C, f)\}
\]

The observation is that when a confident co-location rule can be generated from a co-location pattern then at least one spatial feature in it will have high participation ratio, even though the participation index of whole pattern is low. A high \( maxPI \) is an indication that there are some features that strongly imply rest of the features of the pattern.

\( maxPI \) does not satisfy monotonicity property. This can be verified with a simple example. But it does satisfy weak-monotonicity property defined below.

**Weak-Monotonicity Property:** Given a size-\( k \) co-location pattern \( C \), there exists at most one size-(\( k - 1 \)) sub-pattern \( C' \) such that \( C' \subset C \) and \( maxPI(C') < maxPI(C) \). And so if \( C \) is confident based on \( maxPI \), then so are its at least \( k - 1 \) of size-(\( k - 1 \)) subsets.

In the above example, \textit{cadmium contaminated water} has maximum participation ratio and it forms the \( maxPI \) of co-location pattern \( C \). There are two subsets of size-2 of the given pattern that contain cadmium contaminated water. Because of the monotonicity property of participation ratio, these two subsets must have \( maxPI \) greater than or equal to that of \( C \). The subset that does not contain \textit{cadmium contaminated water} can have \( maxPI \) less than that of \( C \). And hence there can be at most one subset of size-2 with \( maxPI \) less than that of \( C \).

The algorithm works as follows:

1. Applying the weak-monotonicity property in reverse direction, if at least \((k - 1)\) of size-(\( k - 1 \)) patterns have \( maxPI \geq min\_conf \), then the size-\( k \) pattern may also have \( maxPI \geq min\_conf \). Then it forms
a potential candidate. But verifying all \( k-1 \) subsets before generating then candidate of size-\( k \) is very expensive. So the algorithm uses a heuristic to generate candidate confident patterns.

**Heuristic:** If \( C \) is a confident size-\( k \) pattern, then there exist two size-\( (k-1) \) patterns \( C_1 \) and \( C_2 \) that are subsets of \( C \) such that they differ in at most one of the last two features and both \( C_1 \) and \( C_2 \) are confident.

For example, if \( \{A, B, C\} \) and \( \{A, B, D\} \) are confident based on \( maxPI \), then \( \{A, B, C, D\} \) is a candidate to have \( maxPI \geq minconf \).

2. In this way the algorithm generates the candidate confident patterns and then computes their actual \( maxPI \) by generating table instances using any of the methods discussed in previous approaches.

3. Given minimum confidence threshold(\( minconf \)) for co-location rules, the confident co-location pattern is generated from the candidate patterns by finding those patterns whose \( maxPI \) is greater than or equal to \( minconf \). The algorithm uses the actual \( maxPI \) calculated in above step to prune the algorithm based on \( minconf \). All those patterns that pass this filter form the confident co-location patterns.

4. Once the confident pattern \( C = \{f_1, \ldots, f_k\} \) is identified, it is sorted in the descending order of \( maxPI \). Suppose in the sorted set, for \( 1 \leq i \leq l \leq k \), \( pr(C, f_i) \geq minconf \). Then a confident rule \( f_1 \ldots f_l \rightarrow f_{l+1} \ldots f_k \) can be generated from it. This rule says that when a feature \( f_i \) such that \( 1 \leq i \leq l \) is observed at a location, then the probability of observing all other features \( C - \{f_i\} \) in the neighborhood is at least \( maxPI(C) \). The algorithm generates such rules from all confident co-location patterns obtained in previous step.

This approach proposes a way to generate confident co-location rules that may be missed by previous approaches because the co-location pattern may not be prevalent. This situation arises because of the bias of participation index measure towards the frequent features in a co-location pattern. The approach does not require any specification of prevalence threshold from the user and enumerates confident co-location rules using maximal participation index and its properties. The algorithm scales well with number of instances.

### 3.8.2 Finding N-Most Prevalent Co-located Event Sets/Patterns

The first five approaches discussed necessitate the specification of minimum prevalence threshold by the user. However it is difficult for the user to de-
termine what forms a good threshold value all the time. Choosing a large threshold results in small number of results or even no result. Choosing a small threshold generates many results leading to very high computation time and necessitating further filterings. Moreover using constant prevalence threshold for all sizes of co-location candidates generates more small co-locations and less large co-locations. The approach discussed in [10] presents an alternate way to do away with prevalence threshold. This approach finds $N$-most prevalent co-location patterns per each pattern size (up to a certain $k_{\text{max}}$), where $N$ is the desired number of co-location patterns with highest prevalence measure values.

In the worst case, we need to check all possible subsets of given features to generate $N$-most prevalent Co-location patterns. That is, if there are $n$ feature types, then we need to check $2^n - n - 1$ subsets excluding the empty set and size-1 sets from the total subsets. The algorithm proposed under this approach efficiently prunes the search space so as to focus only on potential patterns. The detailed explanation along with examples and required definitions is given below.

**Definition 1:** Let $L$ be a list of all size-$k$ co-location patterns sorted in descending order based on their participation indices. Let $p$ be the participation index on $N$th size-$k$ co-location pattern in $L$. The **$N$-most prevalent $k$-colocation patterns** are a set of size-$k$ co-location patterns having participation index $\geq p$.

The **$N$-most prevalent colocated event sets** are the union of $N$-most prevalent $k$-co-location patterns for $2 \leq k \leq k_{\text{max}}$, where $k_{\text{max}}$ is the upper bound on size of co-location patterns of interest.

**Definition 2:** Given a spatial instance $i_i \in S$, the **neighborhood transaction** of $i_i$ is defined as a set of instances $\{i_i, i_j \in S| R(i_i, i_j) = \text{true} \land i_i^s \text{featuretype} \neq i_j^s \text{featuretype}\}$ where $R$ is a neighbor relationship.

For example for the graph in figure 3, the neighborhood transaction of $A.2$ is \{A.2, B.1, B.2, C.1, C.2\}. Here A.2 is called reference feature. The algorithm generates all neighborhood transactions from the input data set. This neighborhood transaction set does not duplicate or miss any neighborhood information from the data set. The neighborhood transaction set for our example is enumerated in figure 12.

**Definition 3:** The set of distinct features in the neighborhood transaction is called an **event neighborhood transaction**.

Using the neighborhood transaction set generated above, event neigh-
Neighborhood transactions are generated as shown in figure 13. The size of event neighborhood transaction set is usually smaller than that of neighborhood transaction set.

**Definition 4:** An *event tree* or *reference event pattern tree* is a form of prefix tree with properties: (1) Root labeled as a reference feature type, a set of event prefix subtrees as the children of the root, (2) Each node has three fields: event-type, count and node-link where event-type denotes an event the node represents, and count implies the number of event neighborhood transactions represented by the portion of the path reaching this node. Node-link links to the next node in the tree carrying same event type.

For example figure 14 shows the event trees of all three features in figure 3. Here the event transactions starting with B are used to generate event tree of B in a way similar to prefix tree generation.

Figure 15 shows the event sets generated from event trees in figure 14. The resultant sets are called *star instances* since all elements of it form neighbors with the first element. The figure also shows *support* of each set which represents the number of times that its first item event has a neighbor relationship with other events in the set. In figure 15 for event set \{A, B, C\}, in the neighborhood transaction set in 12, we can see two transactions out of four instances of A and hence its support $\frac{2}{4}$. This can be compute from
the count field in the event tree nodes.

Figure 13: Event Neighborhood Transactions

<table>
<thead>
<tr>
<th>Transaction No.</th>
<th>Events</th>
<th>Neighbor Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>2</td>
<td>A</td>
<td>B, C</td>
</tr>
<tr>
<td>3</td>
<td>A</td>
<td>B, C</td>
</tr>
<tr>
<td>4</td>
<td>A</td>
<td>C</td>
</tr>
<tr>
<td>5</td>
<td>B</td>
<td>A, C</td>
</tr>
<tr>
<td>6</td>
<td>B</td>
<td>A, C</td>
</tr>
<tr>
<td>7</td>
<td>B</td>
<td>A, C</td>
</tr>
<tr>
<td>8</td>
<td>C</td>
<td>A, B</td>
</tr>
<tr>
<td>9</td>
<td>C</td>
<td>A, B</td>
</tr>
<tr>
<td>10</td>
<td>C</td>
<td>A</td>
</tr>
</tbody>
</table>

Figure 14: Event Trees

The star candidates thus generated are combined to generate clique candidate set which can have at least one co-location instance. Figure 16 above shows an example of combined clique candidate set. The co-location candidate inherits the frequency from each of its star candidates. For example in figure 16, candidate \{A, B, C\} inherits support values from star candidates \{A, B, C\}, \{B, A, C\} and \{A, B, C\}. These are referred to as upper bound participation ratios, the minimum of whose becomes upper bound participation index of the candidate set.

To further prune the candidates, the algorithm uses the following Lemma:

Lemma 1: If \( \theta \) is the smallest participation index in the result set of N-most
prevalent $l$-colocated event sets($result_l$) and $\theta$ is the participation index of a N-Most prevalent $k$-colocated event set where $k < l$, then $\theta \geq \theta_l$ where $k < l$, provided there are at least $N$ different event types in $result_l$.

Once the cliques candidate sets with upper bound participation indices are obtained (which form the pre-processing), the algorithm works as follows:

1. The algorithm starts generating N-most prevalent colocated event sets from size $k_{max}$ and uses the above lemma to prune candidates at each lower level. All those candidates of size-$(k - 1)$ whose upper bound participation index is less than the least true participation index of the set of N-most colocated event set of size-$k$ are pruned.

2. Then for each of the remaining candidates, true participation index is computed from instances and N-most prevalent sets are enumerated.

The algorithm uses an optimization here for improving performance by eliminating unnecessary enumeration of instances of a candidate co-location. Once we enumerate the first N-prevalent colocated sets of any size-$k$, if the candidate sets obtained after above pruning are ordered by upper bound participation index value and if any of the next candidates has upper bound participation index less than the least true participation index of already enumerated N-prevalent colocated event sets, then the rest of candidates after that candidate can be pruned without enumerating their instances to compute true participation index. This
is because that all the candidates of size-$k$ are ordered by their upper bound participation index values and so if for any of the candidate the upper bound participation index is less than the least true participation index of already enumerated sets, then the candidates that follow it will have their true participation index values certainly less than all those N-prevalent event sets that are already enumerated and hence can be ignored.

3. In order to generate the co-location instances from star instances to compute true participation index of candidate co-locations, we proceed in a way similar to that of join-less approach. The star instances of each instance are obtained from the neighborhood transaction set shown in figure 12. For example for the candidate co-location \{A, B, C\}, if \{A.3, B.3, C.2\} is a star instance, then we have to look up the star instance of B.3 to check if C.2 is neighbor of B.3. If so, then \{A.3, B.3, C.2\} forms an actual co-location instance of \{A, B, C\}.

The approach does away with the necessity of user specifying the minimum prevalence threshold. It develops a new framework for generating top-N most prevalent co-location patterns of each size up to certain threshold size $k_{max}$.

The algorithm presented in this approach shows an efficient way of pruning the search space of all subsets of feature types so that only those subsets which are potential candidates are considered.

**Analysis**

The algorithm discussed in this approach shows increased execution time with increase in number of instances, number of features, distance threshold to measure neighborhood, N and $k_{max}$ because of corresponding increase in candidate co-locations to be examined and co-location instances to be generated.

**3.9 Comparative Analysis**

The first five approaches discussed above attempt to generate co-location instances efficiently and can be compared on the basis some common parameters. The description of parameters and the effect of them on various approaches is detailed below:

1. **No. of Features:** This parameter indicates the number of feature types in the given data set. As the number of feature types increase,
the candidate co-locations and maximum length of co-location patterns will increase. When number of features increase with total number of instances remaining constant, the average number of instances per feature decreases. So the total number of co-location instances will decrease.

As this parameter increases the number co-location instances will decrease and so the number of join computations needed will decrease. Hence the execution time of join-based approach will also decrease.

For the same reason, the execution time of partial-join approach will also decrease. But its execution time is always lower than that of join-based approach because of less number of join computations required in this approach (only for inter-X instances).

The execution time of join-less approach will also decrease and it always performs better than join-based and partial-join approaches because there are no join computations in this approach but only lookups and filtering.

CPI-tree approach will also have decreased execution time but it performs better than all previous approaches because of no joins requirement as well as no extra filtering needed for star instances as in case of join-less approach.

i-CPI tree approach will also show decrease in the execution time and it performs better among all five approaches because it is join-less, no necessity of filtering of star instances and it makes use of apriori property for candidate co-location pruning.

2. **No. of Instances:** This parameter indicates the number of spatial instances in the given data set. As the number of instances increase with features remaining constant, the number of co-location instances will increase.

Since the number of co-location instances increases, the execution time of all five approaches increase. The execution time of join-based approach is the worst of all because of increase in number of join computations needed as the co-location instances increase.

The partial-join approach though has increased execution time because of increased number of joins performs better because of less number of join computations required compared to join-based approach.

The join-less approach will do better than both of above approaches because of no join computations needed despite increase in number of
co-location instances. Only the lookups and filterings will increase and they take less time compared to join computations. But its execution time will increase with number of instances.

The CPI-tree approach will have increased execution time because of increase in co-location instances which necessitates increased traversals on CPI-tree but the performance would be superior compared to all three approaches above because of no necessity of join computations and filtering of star instances.

The i-CPI tree approach will perform better among all approaches because no requirement of joins, filtering of star instances and efficient use of apriori property unlike CPI-tree approach. However its execution time increases because of increase in number of co-location instances which necessitates more traversals on i-CPI tree.

3. **Density:** This parameter indicates the number of instances in a given area. As the data set becomes dense the number of neighborhood relationships increase and hence the number of co-location instances also increase.

As the number of co-location instances increase, because of the same reasons discussed in previous parameter all five approaches will have increased execution time. As in that case the improving order of performance of approaches is: join-based approach, partial-join approach, join-less approach, CPI-tree approach and i-CPI tree approach.

4. **Prevalence Threshold:** This parameter indicates the minimum prevalence threshold value against which the candidate co-locations are pruned. As the prevalence threshold increases, the number of prevalent co-locations and number of co-location instances decrease.

In this case as the number of co-location instances decrease because of increased prevalence threshold, the execution times of join-based, partial-join, join-less, i-CPI tree approaches will decrease as they use prevalence threshold to prune the candidate co-locations. Their comparative performance will improve in the order: join-based, partial-join, join-less, i-CPI tree approaches for the reasons discussed in the first parameter.

However the CPI-tree approach will remain unaffected by the increase in prevalence threshold as this approach will not make use of this parameter to prune candidates co-locations using apriori property.
5. **Distance Threshold:** This parameter indicates the distance threshold to decide if two instances are neighbors. As the distance threshold increases, the number of co-location instances will increase.

As the number of co-location instances increase, the execution times of all approaches will increase. But the improving order of performance of approaches as always is: join-based, partial-join, join-less, CPI-tree and i-CPI tree. This order is because of the same reasons discussed for the second parameter.

6. **No. of Cut-relations:** This parameter indicates the number of cut-relations when the instances are partitioned into clique instances as in the case of partial-join approach. In this approach as the number of cut-relations increase the number of joins required to be performed will increase.

The increase in number of cut-relations will not affect any of the approaches except partial-join approach. As the number of cut-relations increase, the number of inter-X instances will increase in partial-join approach. And so the number of required join computations will increase thereby increasing its execution time.

The following table 3.1 summarizes the affect of increase of each of the above discussed parameters on the execution time of all five approaches under consideration.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Join-Based</th>
<th>Partial-Join</th>
<th>Join-less</th>
<th>CPI-Tree</th>
<th>i-CPI Tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of Features</td>
<td>Decreases</td>
<td>Decreases</td>
<td>Decreases</td>
<td>Decreases</td>
<td>Decreases</td>
</tr>
<tr>
<td>No. of Instances</td>
<td>Increases</td>
<td>Increases</td>
<td>Increases</td>
<td>Increases</td>
<td>Increases</td>
</tr>
<tr>
<td>Density</td>
<td>Increases</td>
<td>Increases</td>
<td>Increases</td>
<td>Increases</td>
<td>Increases</td>
</tr>
<tr>
<td>Prevalence Threshold</td>
<td>Decreases</td>
<td>Decreases</td>
<td>Decreases</td>
<td>Unaffected</td>
<td>Decreases</td>
</tr>
<tr>
<td>Distance Threshold</td>
<td>Increases</td>
<td>Increases</td>
<td>Increases</td>
<td>Increases</td>
<td>Increases</td>
</tr>
<tr>
<td>No. of Cut-relations</td>
<td>NA</td>
<td>Increases</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
</tbody>
</table>
Chapter 4

Open Problems:

All the approaches discussed in this report deal with only point data. A natural extension would be co-location rule mining for line and polygon data. The application of co-location rule mining for line data is in transportation domain where the co-existence of specific types of roads need to be predicted. For example highways and frontage roads co-exist often. By identifying such co-locations, we can evaluate in-vehicle navigation technologies. Main challenges in this extension include defining and enumerating neighborhood relationships among line and polygon objects as well as controlling corresponding geometric costs. The Euclidean distance definition to enumerate neighborhood relationships is not directly applicable to line and polygon data because there is no single point defined to check the distance threshold. If the distance threshold is measured over center points of polygons, it boils down to point objects and cannot effectively represent polygon data properties. Topological definitions for neighborhood may be considered for such data. For example, two lines are said to be neighbors if they intersect and two polygons are said to be neighbors if they touch each other. Buffer-based methods are proposed to extend co-location mining to line and polygon objects.

Another open problem involves approximate co-location mining algorithms using probabilistic approaches. All the approaches discussed in this report provide exact algorithms. Approximate approaches save significant computation time by not performing all the data processing as in case of exact approaches. The approximate algorithms will not generate all the rules unlike exact approaches. They generate only top-k confident rules certain confidence (with some probability of rule being correct and confident) and some tolerance for error. Challenges in this include defining what level of tolerance is allowed and if some false approximations are done for candidate co-locations, they may be pruned and so all higher order candidate co-location will be lost. In this way approximations at each level will have
cascading effects. Though significant research has been done on generation co-location instances out of uncertain data using probabilistic ways, approximate algorithms for co-location rule mining of certain data is not studied to the best of our knowledge.

Another interesting extension would be to mine spatio-temporal co-location patterns. The algorithms discussed in this report will not handle temporal changes. In many cases a given object co-exists with different objects at different times. These are of interest because they find applications in many fields such as army strategies and game design. Mining spatio-temporal co-location patterns is complicated because of no objective way of incorporating temporal dimension in traditional spatial data mining. Much research has happened over the years into this area and many efficient methods such as MDCOPs, STCP-Miner etc. were proposed.

One other interesting extension would be to consider categorical and continuous spatial features instead of boolean spatial features as in above discussed approaches. This is of interest because most of the real world features are not boolean. We may like to find if the co-existing objects vary with finite or infinite set of values an object takes. For example, with the increasing depth of gas basins what chemicals are likely to co-exist in the earth. It is difficult to extend above methods for non-boolean especially continuous spatial features because they need different interestingness measures. Some work has been done in this aspect for extending spatial co-location mining for continuous data.
References


