TR 02-019

The Multi-resolution Co-location Miner: A New Algorithm to Find Co-location Patterns in Spatial Dataset

Shashi Shekhar and Yan Huang

May 01, 2002

The Multi-resolution Co-location Miner: A New Algorithm to Find Co-location Patterns in Spatial Dataset [*]

Shashi Shekhar
Computer Science Department
University of Minnesota
200 Union Street SE
Minneapolis, MN-55455,USA
shekhar@cs.umn.edu

Yan Huang
Computer Science Department
University of Minnesota
200 Union Street SE
Minneapolis, MN-55455,USA
huangyan@cs.umn.edu

ABSTRACT

Given a collection of boolean spatial features, the co-location pattern discovery process finds the subsets of features frequently located together. For example, the analysis of an ecology dataset may reveal the frequent co-location of a fire ignition source feature with a needle vegetation type feature and a drought feature. The spatial co-location rule problem is different from the association rule problem. Even though boolean spatial feature types (also called spatial events) may correspond to items in association rules over market-basket datasets, there is no natural notion of transactions. This creates difficulty in using traditional measures (e.g. support, confidence) and applying association rule mining algorithms which use support based pruning. In our recent work [22], we proposed a notion of user-specified neighborhoods in place of transactions to specify groups of items in [22], new interest measures for spatial co-location patterns which are robust in the face of potentially infinite overlapping neighborhoods,and an algorithm to mine frequent spatial co-location patterns and analyzed its correctness, and completeness. The *Co-location Miner* generates candidate prevalent co-locations in the spatial feature level and generates table instances for the candidate co-locations to check their prevalence. When the false candidate prevalent co-location set is large, the performance of the *Co-location Miner* decreases. Due to spatial autocorrelation, the locations of individual spatial features of a point data set are often clustered spatially, the *Co-location Miner* is computationally expensive without taking spatial autocorrelation into consideration. In this paper, a new algorithm called *Multi-resolution Co-location Miner* is presented. The proposed algorithm has two logical phases, namely filter and refinement. The filter phase summarizes the original point dataset into a smaller lattice dataset using space partitioning which allows the computation of the upper bounds of the interest measures. It eliminates many non-interesting co-locations, reducing the set of candidates to be explored by the refinement phase, which computes the true values of the interest measures. We show that the proposed algorithm is correct and complete and the proposed algorithm is several times faster than the traditional *Co-location Miner* algorithm in a dataset with spatially autocorrelation by experiments.

[*]Supported in part by the Army High Performance Computing Research Center under the auspices of Department of the Army, Army Research Laboratory Cooperative agreement number DAAH04-95-2-0003/contract number DAAH04-95-C-0008

Keywords

spatial data mining, Geographic Information System, spatial co-location rules, association rules

1. INTRODUCTION

Widespread use of spatial databases [7, 20, 21, 28] is leading to an increasing interest in mining interesting and useful but implicit spatial patterns [6, 12, 16, 19, 26]. Efficient tools for extracting information from geo-spatial data, the focus of this work, are crucial to organizations which make decisions based on large spatial datasets. These organizations are spread across many domains including ecology and environmental management, public safety, transportation, public health, business, travel and tourism [3, 11, 14, 8, 23, 26, 29]. We will focus on the application domain of ecology, where scientists are interested in finding frequent co-occurrences among boolean spatial features, e.g. drought, El Nino, substantial increase in vegetation, substantial drop in vegetation, and extremely high precipitation.

Association rule finding [10]is an important data mining technique which has helped retailers interested in finding items frequently bought together to make store arrangements, plan catalogs, and promote products together. Spatial association rules [13] are spatial cases of general association rules where at least one of the predicates is spatial. Association rule mining algorithms [1, 2, 9] assume that a finite set of disjoint transactions are given as input to the algorithms. In market basket data, a transaction consist of a collection of item types purchased together by a customer. Algorithms like *apriori* [2] can efficiently find the frequent itemsets from all the transactions and association rules can be found from these frequent itemsets.

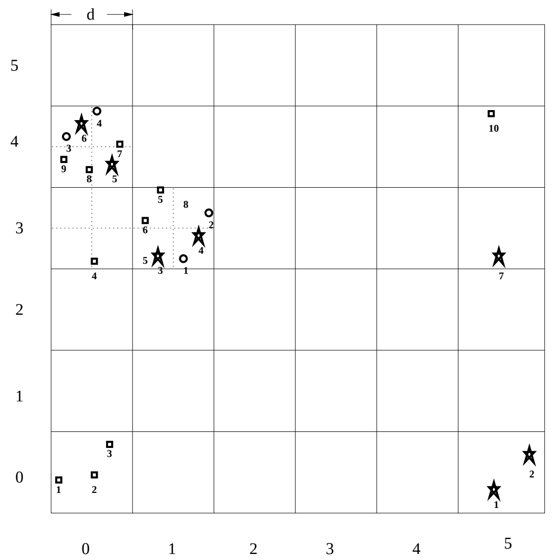


Figure 1: Example dataset

Many spatial datasets consist of instances of a collection of boolean spatial features (e.g., drought, needle leaf vegetation). Figure 1 shows the frequent co-occurrences of some spatial feature types represented by different shapes. While boolean spatial features can be thought of as item types, there may not be an explicit finite set of transactions, due to the continuity of the underlying space. The grid cannot be used to transactionize the data as described in [22]. If spatial association rule discovery is restricted to a reference feature (e.g., city) [13], then transactions can be defined around the instances of this reference feature. Generalizing this paradigm to the case where no reference feature is specified is non-trivial. Defining transactions around locations of instances of all features may yield duplicate counts for many candidate associations. Defining transactions by partitioning space independent of data distribution is an alternative. However, imposing artificial transactions via space partitioning often undercounts instances of tuples intersecting the boundaries of artificial transactions or double-counts instances of tuples co-located together.

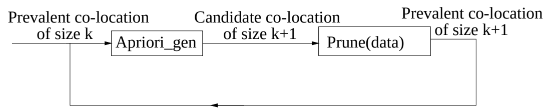
1.1 Related Work and Our Contributions

Approaches to discover co-location rules in the literature can be categorized into two classes, namely spatial statistics and association rules. Spatial statistics-based [4, 5] approaches use measures of spatial correlation to characterize the relationship between different types of spatial features. Measures of spatial correlation include chi-square tests, correlation coefficients, cross-k function, and regression models, as well as their generalizations using spatial neighborhood relationships. Computing spatial correlation measures for all possible co-location patterns can be computationally expensive due to the exponential number of candidates given a large collection of spatial boolean features.

Association rule-based approaches [13] focus on the creation of transactions over space so that an *apriori* like algorithm [2] can be used. Some practitioners use ad-hoc windowing to create transactions, leading to problems of under counting or over counting in the determination of prevalence measures, e.g., support. Another approach is based on the choice of a reference spatial feature [13] to mine all association rules of the following form:

$$is\_a(X, large\_town) \wedge adjacent\_to(X, sea) \Rightarrow close\_to(X, us\_boundary)(80\%) \quad (1)$$

where at least one of the predicates is a spatial predicate. Users are first asked to specify the spatial features which interest them in a form that specifies the reference spatial feature and the relevant spatial features. An example is finding within the map of British Columbia the strong spatial association relationships between large towns and other "near_by" spatial features including mines, country boundary, water and major highways. The large town is the reference spatial feature while the mines, country boundary, etc. are the relevant spatial features. A transaction with items from mines, country boundary, water and major highways is created near_by each large town. The rules found are all related to the large towns such as the example given above. In a sense every rule is stating some facts about large towns, read as for a large town if it has property A then it has property B with some confidence. This approach does not find more general co-location patterns involving no reference spatial feature on the left-hand side of the association rules. For example, consider the co-location pattern of (drought, pine-needle-vegetation) being in a neighborhood implying high probability of a fire-ignition event.


Prevalent co-location of size k
Apriori_gen
Candidate co-location of size k+1
Prune(data)
Prevalent co-location of size k+1
Figure 2: Co-location Miner Algorithm Illustration

The *Co-location Miner* algorithm [22] is an iterative algorithm. In each iteration, it performs two steps, namely feature level *apriori_gen* and instance level pruning, as shown in Figure 4. The first step takes prevalent co-locations of size k and generates candidate co-locations of size $k+1$. The second step analyzes the instance level dataset to compute the interest measures and prunes the candidates based on thresholds on the interest measures to generate prevalent co-locations of size $k+1$. Interest measure computation for a candidate co-location involves spatial joins among instances of the relevant spatial features using a given neighborhood relationship. Interest measure computation for co-locations is much more expensive than that for conventional association rules, due to the lack of transactions. The *Co-location Miner* algorithm incorporated ideas like the use of multiple strategies (e.g. spatial, combinatorial) for computing interest measures as well as reuse of partial results across different co-locations. Despite these optimization, *Co-location Miner* is computationally expensive relative to algorithms for mining association rules. There is a strong need for more efficient algorithms for mining co-location rules.

Contributions: In this paper we use a multi-resolution approach called *Multi-resolution Co-location Miner* to mining co-locations. In each iteration it uses the results from the previous iteration and has two logical phases, namely the filter phase and the refinement phase. The filter procedure summarizes the original point dataset into a smaller lattice dataset using space partitioning. It computes the upper bounds of the values of the interest measures for candidate co-locations using the smaller summary dataset and quickly eliminates candidates using thresholds. Remaining candidates are examined by the refinement phase, which computes exact values of the interest measures using the original dataset to determine the final answers.

We provide a theoretical analysis of the proposed algorithm and experimental performance evaluation results. The theoretical analysis shows that the proposed algorithms are correct and complete. The proposed algorithms are more efficient than the *Co-location Miner*, when instances of individual spatial features are strongly clustered in space and the false candidate co-location rate is high. The experimental results supports the theoretical evaluation by showing the performance superiority of *Multi-resolution Co-location Miner* by 1 to 5 times.

1.2 Basic Concepts and Problem Formulation

The spatial co-location problem looks similar but in fact is very different from the association rule mining problem because of the lack of transactions. In market basket data sets, transactions represent sets of item types bought together by customers. The purpose of mining association rules is to identify frequent item sets for planning store layouts or marketing campaigns. In the spatial co-location rule mining problem, transactions are often not explicit. The transactions in market basket analysis are independent of each other. Transactions are disjoint in the sense that they do not share instances of item types. In contrast, the instances of spatial features are embedded in a space and share a variety of spatial relationships (e.g. neighbor) with each other.

We summarize the key concepts defined to solve the above problem in this section. A **co-location** is a subset of boolean spatial features. A **co-location rule** is of the form: $C_1 \rightarrow C_2(p, cp)$ where C_1 and C_2 are co-locations, p is a number representing the prevalence measure and cp is a number measuring conditional probability. The prevalence measure and the conditional probability measure, called interest measures, need to be defined in spatial application domains and models of interpretation. The **reference feature centric model** enumerates neighborhoods to "materialize" a set of transactions around instances of the reference spatial feature as described in the related work. With "materialized" transactions, the support and confidence of the traditional association rule problem [2] may be used as prevalence and conditional probability measures. The **event centric model** is relevant to applications like ecology, where there are many types of boolean spatial features. Ecologists are interested in finding subsets of spatial features likely to occur in a neighborhood around instances of given subsets of event types. Neighborhood is an important concept in the event centric model. The definition of a neighbor relation is an input and is based on the semantics of the application domains. A neighbor relation may be defined using topological relationships (e.g. connected, adjacent), metric relationships (e.g. Euclidean distance) or a combination (e.g. shortest-path distance in a graph such as road-map). Here we introduce several concepts necessary for defining prevalence and conditional probability measures for spatial datasets in event centric view. $I = \{i_1, \ldots, i_k\}$ is a **row instance** of a co-location $C = \{f_1, \ldots, f_k\}$ if i_j is an instance of feature $f_j (\forall j \in 1, \ldots, k)$ and I has pairwise interested neighboring relationships. For example, {3,1} is an instance of co-location $\{\star, \circ\}$ in Figure 1 where 3 and 1 are unique instance IDs inside each spatial feature type and two instances are neighbors if they are in any square of size $d \times d$. The **table instance** of a co-location $C = \{f_1, \ldots, f_k\}$ is the collection of all its row instances. The **participation ratio** $pr(C, f_i)$ for feature type f_i of a co-location $C = \{f_1, f_2, \ldots, f_k\}$ is the fraction of instances of f_i which participate in any row instance of co-location C. This ratio can be formally defined as $\frac{|distinct(\pi_{f_i}(all\, row\, instances\, of\, C))|}{|instances\, of\, \{f_i\}|}$ where π is a relational projection operation. For example, in Figure 1, instances of co-location $\{\star, \circ\}$ are {3, 1},{3,2},{4,1},{4,2}, {5,3}, {5,4},{6,3}, and {6,4}. 4 instances (3,4,5,and 6) of spatial feature $\star$ out of 7 participates in co-location $\{\star, \circ\}$. So $pr(\{\star, \circ\}, \star) = \frac{4}{7}$. The **participation index** of a co-location $C = \{f_1, f_2, \ldots, f_k\}$ is $min\{pr(C, f_i)\}, i = 1, \cdots, k$. We use the minimal value instead of the product of the participation ratios [22] because minimal value is more meaningful to end users. In Figure 1, participation ratio pr($\{\star, \circ\}, \star$) of feature A in co-location $\{\star, \circ\}$ is $\frac{4}{7}$ as calculated above. Similarly $pr(\{\star, \circ\}, \circ)$ is 1.0. The participation index for co-location $\{\star, \circ\}$ is min{$\frac{4}{7}$, 1.0} = $\frac{4}{7}$. Note that the participation index is monotonically non-increasing with the size of the co-location increasing since any spatial feature that participates in a row instance of a co-location C of size $k+1$ will participates in a row instance of a co-location C' where $C' \subseteq C$. The conditional probability of a co-location rule $C_1 \rightarrow C_2$ in the event centric model is the probability of finding C_2 in a neighborhood of C_1. It can be formally defined as: The conditional probability of a co-location rule $C_1 \rightarrow C_2$ is $\frac{|distinct(\pi_{C_1}(all\, row\, instances\, of\, C_1 \cup C_2))|}{|instances\, of\, C_1|}$ where π is a projection operation.

The lack of transactions in spatial co-location mining creates fundamental differences between association rule mining and co-location rule mining. The major differences are presented in Table 1.

Table 1: Association Rules vs. Co-location Rules

Criteria	Association Rule Mining	Co-location Rule Mining
Item Types	Product types	Spatial Features
Item Collections	Transactions $\{T_i\}$	Neighborhoods
Prevalence $(A \rightarrow B)$	Support: $p(A \cup B \in T_i)$	Participation Index
Conditional Probability $(A \rightarrow B)$	Confidence: $p(B \in T_i \mid A \in T_i)$	Conditional Probability $p(B \in Neighborhood\, of\, L \mid A\, at\, L)$

We formalize the event centric co-location rule mining problem as follows:
Given:
1) a set T of K Boolean spatial feature types T=$\{f_1, f_2, \ldots, f_K\}$

2) a set of N instances P=$\{p_1 \ldots p_N\}$, each $p_i \in P$ is a vector <instance-id, spatial feature type, location> where spatial feature type $\in$ T and location $\in$ spatial framework S
3) A neighbor relation R over locations in S
4) Min prevalence threshold value, min conditional probability threshold
Objectives:
1) **Completeness**: We say an algorithm is complete if it finds all spatial co-location rules which have prevalences and conditional probabilities greater than user specified thresholds.
2) **Correctness**:We say an algorithm is correct if any spatial co-location rules it finds has prevalence and conditional probabilities greater than user specified thresholds.
3) **Computational efficiency**: IO cost and CPU cost to generate the co-location rules should be acceptable
Find:
Co-location rules with high prevalence and high conditional probability
Constraints:
1) R is symmetric and reflexive
2) Monotonic prevalence measure
3) Conditional probability measures are specified by the event centric model

1.3 Outline and Scope

In section 2, we present the *Multi-resolution Co-location Miner* algorithm. Section 3 analyzes the algorithm for their completeness, correctness, and complexity comparisons with the original *Co-location Miner.* The design of an experiment to evaluate the proposed algorithm as well as the experimental results are presented in the section 4

The scope of this paper is limited to co-location rules in two dimensional Euclidean space. Issues beyond the scope of the paper include other spatial patterns, spatio-temporal co-locations, as well as system implementation issues such as selection of index, buffering policy etc.

2. MULTI-RESOLUTION CO-LOCATION MINER

We present a high level description of main steps of the algorithm before providing the details.

Input:
1) K boolean spatial instance types and their instances

$$P = \{< f_i, \{I\} > | f_i \in \{f_1, f_2, \ldots, f_K\}, I \in S\}$$

where S is the set of all interested locations
2) A symmetric and reflexive neighbor relation R
3) A multi-resolution lattice definition to transform the dataset to

$$P_C = \{< f_i, \{< C, \# >\} > | f_i \in \{f_1, f2, \ldots, f_K\}, C \in S_C\}$$

where S_C is the set of all partitions and # is the number of point instances of f_i in C
4) A symmetric and reflexive neighbor relation R_C which is is compatible with R
5) A user specified minimum threshold prevalence measure ($min\_prevalence$)
6) A user specified minimum conditional probability (min_cond_prob)
Output:
Co-location rule sets with partition index $> min\_prevalence$ and
conditional probability $> min\_cond\_prob$
Method:
1) Initialization
2) **for** size of co-locations in (1, 2, 3, ... , $K-1$) **do**
2.1) Schema level (spatial feature level) pruning
2.2) Coarse resolution instance level pruning
2.3) Fine resolution instance level pruning
2.4) Generate co-location rules
3) **end**;

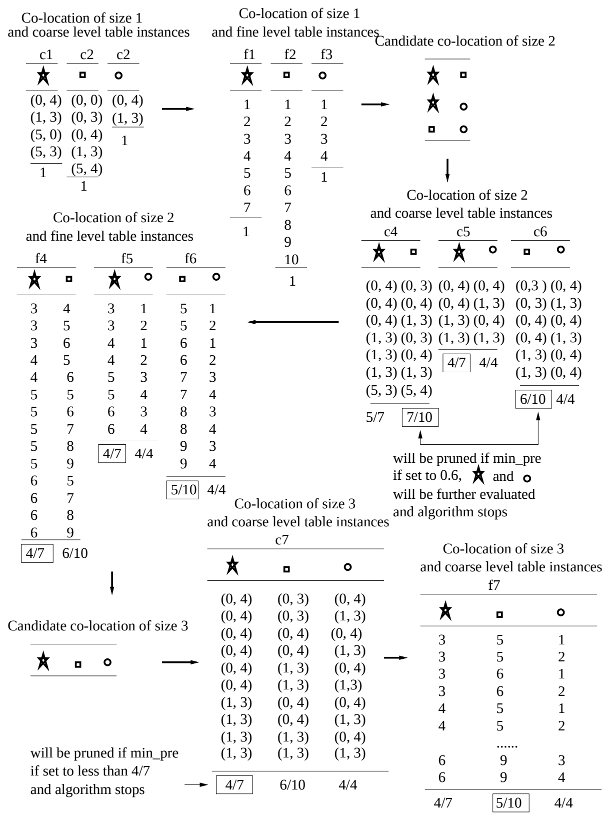
In Input 4), we say R_C is compatible with R if any two partitions are neighbors if any pair of points from each of the two partitions are neighbors in the original dataset P.

Step 2.1 to **Step 2.4** loops through 1 to $K-1$ to generate prevalent co-locations of size 2 or more, iterating on increasing values of sizes of co-locations. Each iteration may use the co-locations and their table instances generated in the previous iteration. The loop breaks whenever an empty co-location set of some size is generated.

Whenever necessary, we use Figure 1 as an example dataset to illustrate algorithms, analysis, and lemma proofs. In Figure 1, different shapes represent different spatial feature types. Two locations are neighbors if they locate in any square of size $d \times d$. Every instance has a unique ID in its spatial feature type and is labeled below it in the figure. A lattice with uniform cell size d is super-imposed on the dataset. In the lattice dataset, two cells are neighbors if their centers are in any square of size $d \times d$. Without considering the edge effects, this is a 9 adjacency neighbor definition including the cell itself. This new neighborhood definition guarantees that two cells are neighbors if any pair of points from each of the two cells are neighbors in the original dataset. Cell ID (and instance in the lattice dataset) is determined by reading the x-axis followed by the y-axis.

Step 1 initializes the the prevalent size 1 co-location set with the input P of the algorithm. At the same time, P_C is needed for generating table instances of candidate co-locations of size 2 in the coarse resolution instance level pruning in step 2.2. Thus size 1 coarse resolution co-location set is initialized with P_C. The participation indexes of singleton co-locations are 1 and all singleton co-locations are prevalent. For example, in Figure 3 co-locations of size 1 in the fine resolution level are represented by tables f_1, f_2, and f_3. Co-locations of size 1 in the coarse resolution level are represented by tables c_1, c_2, and c_3.

Step 2.1 uses *apriori_gen* to generate candidate prevalent co-locations of size $k+1$ from prevalent co-locations of size k and this substep works on spatial feature type and co-location level. The *apriori_gen* is an adoption function from the *apriori* [2]. It takes as argument C_k, the set of all prevalent size k co-locations. The function works as follows. First, in the *join* step, we join C_k with C_k:



Co-location of size 1 and coarse level table instances
Co-location of size 1 and fine level table instances
Candidate co-location of size 2
Co-location of size 2 and coarse level table instances
Co-location of size 2 and fine level table instances
will be pruned if min_pre if set to 0.6, ★ and ○ will be further evaluated and algorithm stops
Co-location of size 3 and coarse level table instances
Candidate co-location of size 3
will be pruned if min_pre if set to less than 4/7 and algorithm stops

Figure 3: Multi-resolution Co-location Miner Algorithm Illustration on Example Database in Figure 1

```
insert into $C_{k+1}$
select $p$.feature$_1$, $p$.feature$_2$, ..., $p$.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$_{k-1}$, $p$.feature$_k$ < $q$.feature$_k$;
```

A sort-merge join is a very competitive candidate for the above query noticing that the results of each iteration is naturally sorted for the next iteration.

Next, in the *prune* step, we delete all candidate co-locations $c \in C_{k+1}$ such that some k-subset of c is not in C_k because of the monotonicity property of the prevalence measure:

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

For example, if the size 2 co-location set is $\{\{\star, square\}, \{\star, \circ\}\}$, the *join* step will produce $\{\{\star, square, \circ\}\}$. The *prune* step will delete $\{\star, square, \circ\}$ from $\{\{\star, square, \circ\}\}$ because $\{square, \circ\}$ is not a prevalent co-location of size 2.

Step 2.2 generates coarse level table instances of candidate co-locations of size $k+1$ which passed the filter of step 2.1. Co-locations with empty table instances will be eliminated from the candidate prevalent co-location set of size $k+1$. This step takes size $k+1$ candidate co-location set C_{k+1} as an argument and works as follows.

```
forall co-location $c \in C_{k+1}$
insert into $T_c$ // $T_c$ is the coarse level table instance of co-location $c$
select $p$.instance$_1$, $p$.instance$_2$, ..., $p$.instance$_k$, $q$.instance$_k$
from $c$.coarse_table_instance_id_1 $p$, $c$.coarse_table_instance_id$_2$ $q$
where $p$.instance$_1$=$q$.instance$_1$, ..., $p$.instance$_{k-1}$=$q$.instance$_{k-1}$, ($p$.instance$_k$, $q$.instance $_k$) $\in R_C$;
end;
```

Here a sort-merge join is again preferred because that the table instances of each iteration is naturally sorted for the next iteration. Then all co-locations with empty table instance will be eliminated from C_{k+1}. For example, in Figure 3, table c_4 of co-location $\{\star, square\}$ and table c_5 of co-location $\{\star, \circ\}$ are joined to produce the coarse level table instance of co-location $\{\star, square, \circ\}$ because co-location $\{\star, square\}$ and co-location $\{\star, \circ\}$ were joined in *apriori_gen* to produce co-location $\{\star, square, \circ\}$ in the previous step. In the example, row instance $\{(0,4),(0,3)\}$ of table c_4 and row instance $\{(0,4),(0,4)\}$ of table c_5 are joined to generate row instance $\{(0,4),(0,3),(0,4)\}$ of co-location $\{\star, square, \circ\}$ (Table c_7).

Then **Step 2.2** also calculates the over-estimated participation indexes for all candidate co-locations based on the coarse level table instances and prunes the co-locations using the given prevalence threshold. Computation of the over-estimated participation index for a co-location C based on the coarse level table instance of C is similar to the corresponding procedure in **Step 2.3**. It requires scanning of its coarse level table instance to compute over-estimated participation ratios for each feature in the co-location. This computation can be modeled as a *project-unique* operation on columns of the coarse level table instance of C. This can be accomplished by keeping a bitmap of size |coarse level instance of f_i| for each feature f_i of co-location C. One scan of the table instance of C will be enough to put 1s in corresponding bits in each bitmap. The following part is different from the corresponding procedure in **Step 2.3**. Here we could not just add up all the 1s in the bitmap because each 1 in the bitmap is a partition with includes a set of point instances. For each spatial feature f_i, we add up all the count of point instances in each coarse level instance with 1s in its corresponding bitmap (p_{f_i}) and divide it by |instance of f_i| to get the over-estimated participation ratio of feature f_i. For example in Figure 3, to calcuate the over-estimated participation indexes for co-location $\{\star, square, \circ\}$, we first calculate the participation ratios by scanning table c_7. For feature $\star$ in $\{\star, square, \circ\}$, only (0,4) and (1,3) participate. Both cell (0,4) and cell (1,3) include 2 stars. Totally four stars are estimated to participate in $\{\star, square, \circ\}$ and the participation ratio pr($\{\star, square, \circ\}, \star$) is overestimated as 4/7 where 7 is the total number of stars. Similarly we can get the over-estimated participation ratio for *square* (6/10) and $\circ$ (4/4) and the over-estimated participation index is the min{4/7,6/10,4/4}=4/7. All co-locations with over-estimated participation index less than user specified threshold (false candidates) are eliminated from C_{k+1} without passing to the next step. For example, in Figure 3 if the prevalence threshold is set to be 0.6, $\{\star, square\}$ with overestimated participation index 7/10 and $\{\star, \circ\}$ with overestimated participation index 6/10 will be pruned. Only fine level table instances of $\{\star, \circ\}$ needs to be generated and its actual participation index needs to be calculated.

Step 2.3 generates fine level table instances of candidate co-locations. It works the same as the corresponding procedure in **Step 2.2** except that it works on the fine instance level by joining fine level table instances with a neighborhood R instead of R_C and are likely work in a smaller candidate co-location set which passes filter of step 2.2. For example, in Figure 3, table f_4 of co-location $\{\star, square\}$ and table f_5 of co-location $\{\star, \circ\}$ are joined to produce the fine level table instance of co-location $\{\star, square, \circ\}$.

Then **Step 2.3** also calculates the actual participation indexes for all candidate co-locations in C_{k+1} and prunes the co-locations using the prevalence threshold. This is similar to the corresponding procedure in **Step 2.2**. It requires scanning its fine level table instance to compute participation ratios for each feature in the co-location. This can be accomplished by keeping a bitmap of size |instance of f_i| for each feature f_i of co-location C. One scan of the table instance of C will be enough to put 1s in corresponding bits in each bitmap. By summarizing the total number of 1s (p_{f_i}) in each bitmap, we obtain the participation ratio of each feature f_i (divide p_{f_i} by |instance of f_i|).

Step 2.4 generates all the co-location rules with the user defined *min_prev* and *min_cond_prob*.
For each prevalent co-location C, we enumerate every subset C' of C and calculate the conditional probability measure for the spatial co-location rule: $C' \rightarrow C - C'$. 1) Project the table instance of C on C' to get CC. 2) Calculate the cardinality of CC after duplicate elimination to get the N_p. 3) Divide N_p by the cardinality of C' (which has already been calculated and kept in the previous iterations) to get the conditional probability. 4) Produce: $C' \rightarrow C - C'$ if the conditional probability is above user specified threshold.

Like in *Co-location Miner*, using a geometric approach suffers from a lack of spatial feature level pruning but it keeps the information of nearby regions. The combinatorial approach described above does not keep local information but it benefits from spatial feature level pruning. Since all the participation indexes of singleton co-locations are 1, making all singleton co-locations prevelant and spatial feature level pruning is not effective when generating co-locations of size 2, the geometric approach is preferred in this step. However, when generating co-locations of size 3 or more, spatial feature level pruning is the dominant optimization technique, making the combinatorial method the preferred approach in this step. We integrate the advantages of the geometric and combinatorial approaches by using the geometric approach to generate co-locations of size 2 and switches to combinatorial approach to generate co-locations of size 3 or more in *Multi-resolution Co-location Miner* too.

To compare the *Co-location Miner* and the *Multi-resolution Co-location Miner*, we illustrate both algorithms in Figure 4 and Figure 5. Both graphs illustrate the iterations to generate co-locations of size 3 or more. Table 2 gives the meaning of the parameters of $C(algo, co\_loc\,type, size)$ used in algorithm illustration graphs and later on in the performance comparisons in section 3. In the algorithm illustration graphs, the *apriori_gen* only works in the spatial feature type level, thus only the input co-location set from the previous iteration matters. The **prune** step generates the table instance of each input candidate co-location based on coarse resolution level or the original datasets, calculates the corresponding participation indexes, and prunes based on participation index threshold. The *Muti-resolution Co-location Miner* algorithm illustrated in Figure 5 is similar to the *Co-location Miner*, illustrated in Figure 4, except for one additional pruning in **Step 2.2** which uses coarse level dataset.

Table 2: Parameters of $C(algo, co\_loc\,type, size)$

algo	*Co-location Miner*(cm) or the *Muti-resolution Co-location Miner*(mcm)
co_loc type	the candidate co-locations generated by the *apriori_gen* (cand), the subset of the candidate co-locations generated by *apriori_gen* (sub_cand), the prevalent co-locations(prev)
size	size of the co-location

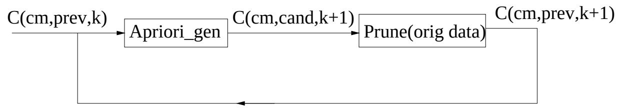


Figure 4: Co-location Miner Illustration

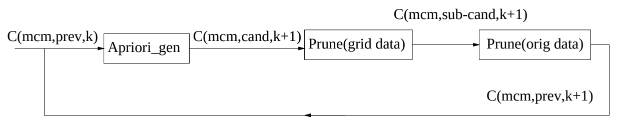


Figure 5: Muti-resolution Co-location Miner Illustration

3. ANALYSIS OF MULTI-RESOLUTION CO-LOCATION MINER

In this section, we analyze the coarse instance level pruning, the completeness and correctness of the *Multi-resolution Co-location Miner*, and its performance compared with the *Co-location Miner*.

3.1 Analysis of the Coarse Instance Level Pruning

In this section, we prove a lemma regarding the over-estimation of participation indexes of **Step 2.2**.

LEMMA 1. *The participation indexes calculated by* **Step 2.2** *of the Multi-resolution Co-location Miner are over-estimates of the participation indexes of the original dataset. The candidate co-location set found by* **Step 2.2** *is a superset of the prevalent co-location set on the original dataset.*

When $co - location\, size = 1$, Lemma 1 is trivially true. Suppose when $co - location\, size = k$ Lemma 1 is true, we consider when $co - location\, size = k+1$. For each candidate co-location C of size $k+1$ generated from the *apriori_gen* by joining C_1 and C_2 of size k, we generate its coarse level instance table. This is achieved by joining the coarse level instance tables of C_1 and C_2. Because lemma 1 is true for co-locations of size k, the candidate co-location set of size k found by **Step 2.2** is a superset of the prevalent co-location set on the original dataset. So C_1 and C_2 are in the candidate co-location set found by **Step 2.2** in the previous iteration and their coarse level table instances are available to be joined to produce the coarse level table instance of C. The correctness and completeness of the table join to produce the coarse level table instance of C are similar to that proved in the *Co-location Miner* using a new neighborhood relationship R_C. Because of the new neighborhood definition R_C if $Neighbor_R(p_1, p_2)$ in the original dataset, then $Neighbor_{R_C}(c_1, c_2)$ in the coarse level dataset given $p_1 \in c_1$ and $p_2 \in c_2$. When we calculate the participation index, any point instance which participates in the co-location in the original dataset will contribute to the counts during the participation ratio calculation. So the **Step 2.2** over-estimates the participation ratios (indexes). After the calculation of participation indexes, the pruning is carried out using the same user given threshold. Because the participation indexes are over-estimated, the pruning will not eliminate any actual prevalent co-location. Thus the candidate co-location set found by **Step 2.2** is a superset of the prevalent co-location set on the original dataset.

For example, in Figure 1, when we calculate the participation ratio of $\star$ in $\{\star, square\}$ the point instances {3,4,5,6} of spatial feature $\star$ are correctly calculated as the instances with at least one instance of spatial feature *square* nearby. But instance 7 of spatial feature $\star$ is calculated as an instance with spatial feature *square* nearby too because the new neighborhood relationship R_C is less stringent than R. Thus the participation ratio 5/7 calculated by the **Step 2.2** is an over-estimate of the real ratio (4/7) over the original dataset.

The **Step 2.2** is much more efficient at finding candidate co-locations and their over-estimated participation indexes in dense datasets compared to **Step 2.3**, which finds the exact co-locations and their participation indexes. The reason is that the **Step 2.2** works on the lattice dataset. A set of clustered points is treated as one instance in the grid dataset, avoiding the exponential combination of the point instances in and between cells. For example, in Figure 1, when we generate the table instance of $\{\star, square, \circ\}$ in the grid dataset, we get 10 row instances as in table c_7. The number of row instances of the fine level table instance of $\{\star, square, \circ\}$ is as many as 20 as in table f_7 even in such a small degree of clusterness. The ratio of the fine level table instance size over the coarse level table instance size is theoretically exponentially increasing with the size of the co-location.

3.2 Completeness of the Multi-resolution Co-location Miner

LEMMA 2. *Algorithm Multi-resolution Co-location Miner is complete.*

Given that we have proved that the *Co-location Miner* algorithm is complete, we only need to prove that the coarse resolution instance level pruning in **Step 2.2** does not affect the completeness. By Lemma 1, the co-location set found by **Step 2.2** is a superset of the prevalent co-location set on the original dataset. Thus **Step 2.2** does not falsely eliminate any possible co-locations and the *Multi-resolution Co-location Miner* algorithm is complete.

3.3 Correctness of the Multi-resolution Co-location Miner

LEMMA 3. *Algorithm Multi-resolution Co-location Miner is correct.*

Adding one more filter to the original *Co-location Miner* algorithm will not affect the correctness of the *Co-location Miner*. Thus the *Multi-resolution Co-location Miner* is correct.

3.4 Computational Complexity Comparison of the Multi-resolution Co-location Miner and the Co-location Miner

Let us represent the cost of iteration k of the *Co-location Miner* algorithm as follows:

$$\begin{aligned} T_{cm}(k) =& T_{apriori\_gen(C(cm,prev,k))} \\ &+ T_{prune(C(cm,cand,k+1),orig\,data)} \end{aligned} \tag{2}$$

The cost of the *Multi-resolution Co-location Miner* is computed as follows:

$$\begin{aligned} T_{mcm}(k) =& T_{apriori\_gen(C(tc,prev,k))} \\ &+ T_{prune(C(mcm,cand,k+1),grid\,data)} \\ &+ T_{prune(C(mcm,sub-cand,k+1),orig\,data)} \end{aligned} \tag{3}$$

Here $T_{apriori\_gen(C(cm,prev,k))} = T_{apriori\_gen(C(mcm,prev,k))}$ because *apriori_gen* works on spatial feature level.

Now we compare the cost of the *Co-location Miner* algorithm and the *Multi-resolution Co-location Miner* algorithm. Notice that the bulk of time is consumed by generating table instances and calculating the participation indexes, the ratio can be simplified as:

$$\frac{T_{mcm}(k)}{T_{cm}(k)} = \frac{T_{prune(C(mcm,cand,k+1),grid\,data)} + T_{prune(C(mcm,sub-cand,k+1),orig\,data)}}{T_{prune(C(cm,cand,k+1),orig\,data)}} \tag{4}$$

Furthermore, we assume that the average time to generate a table instance in the original dataset is $T_{original}$ and the average time to generate a table instance in the grid dataset is T_{grid}. The number of candidate co-locations generated by the *apriori_gen* is $|C_{k+1}|$ and the number of candidate co-locations after the coarse instance level pruning is $|C'_{k+1}|$, equation 4 can be estimated by:

$$\begin{aligned}\frac{T_{mcm}(k)}{T_{cm}(k)} &= \frac{|C_{k+1}| \times T_{grid} + |C'_{k+1}| \times T_{orig}}{|C_{k+1}| \times T_{orig}} \\ &= \frac{T_{grid}}{T_{orig}} + \frac{|C'_{k+1}|}{|C_{k+1}|}\end{aligned} \qquad (5)$$

The first term of the ratio is controlled by the "clumpiness" of the locations of spatial features which means the average number of instances of spatial features clustered together in the space and the second term is controlled by the coarse instance level pruning efficiency. When the locations of spatial features are clustered the sizes of the fine level table instances are much larger than the sizes of the coarse level table instances and the time needed to generate fine level table instances are much more than the time needed to generate coarse level table instances. In our experiments in next section, we use the parameter m_{clump} which controls the number of instances clumping together for each spatial feature to evaluate the first term and the parameter $m_{overlap}$ which represents the possible false candidates ratio to evaluate the second term. From the formula, we can see that the *Multi-resolution Co-location Miner* is more efficient than the *Co-location Miner* when the locations of spatial features are clustered and the false candidate ratio is high.

4. EXPERIMENT DESIGN AND PERFORMANCE EVALUATION

Figure 6 describes the experimental setup to evaluate relative performance of the alternative algorithms. We evaluates the performance of the algorithms with synthetic data generated using a methodology similar to methodologies used to evaluate algorithms for mining association rules [2]. Synthetic datasets allow better control towards studying effects of interesting parameters, e.g. number of core co-locations (N_{co_loc}), expected size (λ_1) of maximal co-locations, etc. The list of the parameters is presented in Table 3. A data-flow diagram of data generation process is shown in Figure 6. The process began with the generation of subsets of spatial features (core co-locations). To generate a subset of features, we first chose the size of the subset from a Poisson distribution with mean (λ_1). Then a set of features for this core co-location pattern was chosen. For each core co-location, $m_{overlap}$ maximal co-locations were generated by appending one more spatial feature to a core co-location. To simplify interpretation of the results, co-locations were kept mutually disjoint. $m_{overlap}$ was used to control the number of possible false candidate co-locations generated by the *apriori_gen*. The larger $m_{overlap}$ is, the more false candidate *apriori_gen* generates. The size of each table instance of each co-location was chosen from another Poisson distribution with mean λ_2. Next, we generated the set of neighborhoods for co-locations using the size of their table instances from the previous step. m_{clump} point locations for each feature in the co-location were embedded inside a neighborhood of size d. The locations of neighborhoods were chosen at random in the overall spatial framework. For simplicity, the shape of the overall spatial framework was a rectangle of size $D_1 \times D_2$ and the size of each neighborhood was $d \times d$. The final step was to add noise. The model for noise used two parameters, namely the ratio of noise features r_{noise_f} and number of noise instances p_{noise_n}. Noise was added by generating a set of instances of features in a set of noise features disjoint with the features involving generation of maximal co-locations and placing those at random locations in the global spatial framework.

The collections of spatial datasets used were generated based on a base dataset shown in Table 4. The base dataset used a rectangle spatial framework of size 500×1000, a square neighborhood of size 10×10, an average co-location size of 5, an average table instance size of 50 when $m_{clump} = 1$ a noise feature ratio of 0.5, a noise number of 1000, a core co-location size of 4, and a overlapping degree of 2. Spatial framework sizes were proportional to the total number of instances to avoid unexpected patterns created by too crowded instances. Then we ranged the overlapping degree ($m_{overlap}$) from 2 to 8 and the clumpiness measure (m_{clump}) from 5 to 20 to generate other datasets. These datasets are used by the co-location algorithm driver module to collect the performance statistics, as shown in Figure 6. The driver calls the candidate algorithm (i.e. the *Co-location Miner* and the *Multi-resolution Co-location Miner*). Each algorithm was instrumented to measure execution time for discovering co-locations of size 2 as well the execution time for co-locations of size 3 or more. The execution time was measured using the "time" utility in the C++ language on a Sun Ultra 10 work station with a 440 MHz CPU, and 128 Mbytes memory running the SunOS 5.7 operating system. The last module in the experimental setup was responsible for summarizing the measurements in the form of plots and tables reported in the following section.

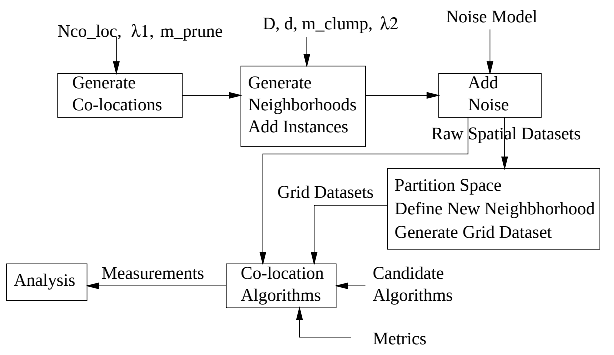


Figure 6: Experimental Setup and Design

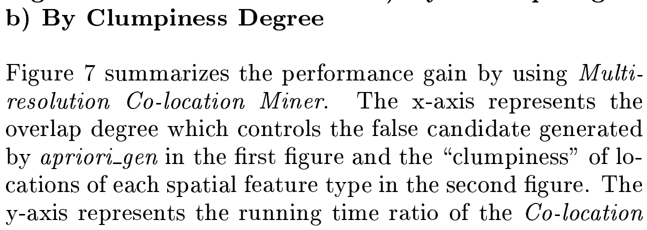


Figure 7: Performance Ratio a) By Overlap Degree b) By Clumpiness Degree

Figure 7 summarizes the performance gain by using *Multi-resolution Co-location Miner*. The x-axis represents the overlap degree which controls the false candidate generated by *apriori_gen* in the first figure and the "clumpiness" of locations of each spatial feature type in the second figure. The y-axis represents the running time ratio of the *Co-location*

Table 3: Parameters Used to Generate Synthetic Data

N_{co_loc}	The number of core co-locations
λ_1	The parameter of the Poisson distribution to define the size the core co-locations
λ_2	The parameter of the Poisson distribution to define the size of the table instance of each co-location when $m_{clump} = 1$
$D_1 \times D_2$	The size of the spatial framework
d	The size of the square to define a co-location
r_{noise_f}	The ratio the of number of noise features over the number of features involved in generating the maximal co-locations
r_{noise_n}	The number of noise instances
$m_{overlap}$	The number of co-location generated by appending one more spatial feature for each core co-location
m_{clump}	The number of instances generated for each spatial feature in a neighborhood for a co-location

Table 4: Synthetic Data Parameters

$D_1 \times D_2$	d	N_{co_loc}	$m_{overlap}$	λ_1	λ_2	r_{noise_f}	r_{noise_n}
500×1000	10	4	2	5	50	.5	1000

Miner over the *Multi-resolution Co-location Miner* when generating co-locations of size 3 or more. The time needed for generate co-location of size 2 is not included because it depends on the spatial join methods chosen and further more is dominated by the time to generate co-locations of size 3 or more. We used the participation index we expected known when generating the datasets as the thresholds. As the overlap degree and the number of false candidates increase, the performance improvements are increasing by 1 to 4.5 times. It improves faster as the "clumpiness" increases. The performance improvements have the similar trend with the increase of the "clumpiness".

Figure 8 summarizes the time ratio of the coarse level pruning and the fine level pruning phases. Similarly, the x-axis represents the overlap degree in the first figure and the "clumpiness" of locations of each spatial feature type in the second figure. In the first figure, the ratio slowly decreases as the overlap degree increases and the trend is faster for smaller clumpiness degree and is between 0.1 to 0.5. In the second figure, the ratio decreases faster than that in the first figure as the clumpiness degree increase. This is because for a fixed overlap degree, with the clumpiness increasing, the time needed to generate coarse level table instances is not increasing as fast as the time needed to generate fine level table instances.

5. CONCLUSION AND FUTURE WORK

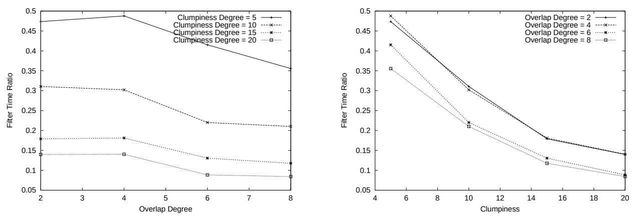


Figure 8: Filter Time Ratio 1) By Overlap Degree 2) By Clumpiness Degree

In this paper, we proposed the *Multi-resolution Co-location Miner* algorithm using a lattice to help calculate the upper bounds for the prevalence measures. We have shown that these algorithms are complete and correct. By theoretical analysis we have shown that when the dataset is clustered and there are some false candidates, the multi-resolution algorithm is expected to be faster than the traditional *co-location miner*. The proposed algorithm uses an innovative approach to deal with exponential increase of instances of co-locations due to the clusterness of spatial datasets. The experimental results confirmed our theoretical analysis. We would also like to explore co-location patterns in lattice datasets which lack initial point dataset information due to various reasons(e.g. security).

Acknowledgments

We thank C.T. Lu, Xiaobin Ma, Hui Xiong, and Pusheng Zhang (University of Minnesota) for their valuable feedback on early versions of this paper.

6. REFERENCES

[1] R. Agarwal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proc. of the ACM SIGMOD Conference on Management of Data, pages 207-216*, may 1993.

[2] R. Agarwal and R. Srikant. Fast algorithms for Mining association rules. *VLDB*, may 1994.

[3] P.S. Albert and L.M. McShane. A Generalized Estimating Equations Approach for Spatially Correlated Binary Data: Applications to the Analysis of Neuroimaging Data. *Biometrics(Publisher: Washington, Biometric Society, Etc.), 1:627-638*, 1995.

[4] Y. Chou. In *Exploring Spatial Analysis in Geographic Information System, Onward Press, (ISBN: 1-56690-119-7)*, 1997.

[5] N. Cressie. In *Statistics for Spatial Data, Wiley-Interscience, (ISBN:0-471-00255-0)*, 1993.

[6] G. Greenman. Turning a map into a cake layer of information. *New York Times*, Feb 12 2000.

[7] R.H. Guting. An Introduction to Spatial Database Systems. In *Very Large Data Bases Jorunal(Publisher: Springer Verlag)*, October 1994.

[8] R.J. Haining. Spatial Data Analysis in the Social and Environmental Sciences. In *Cambridge University Press, Cambfidge, U.K*, 1989.

[9] J. Han and Y. Fu. Discovery of multiple-level association rules from large databases. In *Proc. 1995 Int. Conf. Very Large Data Bases, pages 420-431, Zurich, Switzerland*, September 1995.

[10] J. Hipp, U. Guntzer, and G. Nakaeizadeh. Algorithms for Association Rule Mining - A General Survey and Comparison. In *Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2000.

[11] Issaks, Edward, and M. Svivastava. Applied Geostatistics. In *Oxford University Press, Oxford*, 1989.

[12] K. Koperski, J. Adhikary, and J. Han. Spatial Data Mining: Progress and Challenges. In *Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD'96)*, 1996.

[13] K. Koperski and J. Han. Discovery of Spatial Association Rules in Geographic Informa tion Databases. In *Proc. Fourth International Symposium on Large Spatial Data bases, Maine. 47-66*, 1995.

[14] P. Krugman. Development, Geography, and Economic theory. In *MIT Press, Camgridge, MA*, 1995.

[15] Z. Li, J. Cihlar, L. Moreau, F. Huang, and B. Lee. Monitoring Fire Activities in the Boreal Ecosystem. *Journal Geophys. Res,. 102(29):611-629*, 1997.

[16] D. Mark. Geographical Information Science: Critical Issues in an Emerging Cross-disciplinary Research Domain. In *In NSF Workshop*, February 1999.

[17] D.C. Nepstad, A. Verissimo, A. Alencar, C. Nobre, E. Lima, P. Lefebvre, P. Schlesinger, C. Potter, P. Moutinho, E. Mendoza, M. Cochrane, and V. Brooks. Large-scale Improverishment of Amazonian Forests by Logging and Fire. *Nature, 398:505-508*, 1999.

[18] J.S. Park, M. Chen, and P.S. Yu. Using a Hash-Based Method with Transaction Trimming for Mining Association Rules. In *IEEE Transactions on Knowledge and Data Engineering, vol. 9, no. 5, pp. 813-825*, Sep-Oct 1997.

[19] J.F. Roddick and M. Spiliopoulou. A Bibliography of Temporal, Spatial and Spatio-temporal Data Mining Research. *ACM Special Interest Group on Knowledge Discovery in Data Mining(SIGKDD) Explorations*, 1999.

[20] S. Shekhar and S. Chawla. Spatial Databases: Issues, Implementation and Trends. *Prentice Hall (under contract)*, 2001.

[21] S. Shekhar, S. Chawla, S. Ravada, A. Fetterer, X. Liu, and C.T. Lu. Spatial Databases: Accomplishments and Research Needs. *IEEE Transactions on Knowledge and Data Engineering, 11(1)*, Jan-Feb 1999.

[22] S. Shekhar and Y. Huang. Co-location Rules Mining: A Summary of Results. In *Proc. Spatio-temporal Symposium on Databases*, 2001.

[23] S. Shekhar, T.A. Yang, and P. Hancock. An Intelligent Vehicle Highway Information Management System. *Intl Jr. on Microcomputers in Civil Engineering (Publisher: Blackwell Publishers), 8(3)*, 1993.

[24] R. Srikant and R. Agrawal. Mining Generalized Association Rules. In *Proc. of the 21st Int'l Conference on Very Large Databases, Zurich, Switzerland*, 1997.

[25] R. Srikant, Q. Vu, and R. Agrawal. Mining Association Rules with Item Constraints. In *Proc. of the 3rd Int'l Conference on Knowledge Discovery in Databases and Data Mining, Newport Beach, California*, Aug 1997.

[26] P. Stolorz, H. Nakamura, E. Mesrobian, R.R. Muntz, E.C. Shek, J.R. Santos, J. Yi, K. Ng, S.Y. Chien, R. Mechoso, and J.D. Farrara. Fast Spatio-Temporal Data Mining of Large Geophysical Datasets. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining, AAAI Press, 300-305*, 1995.

[27] C. Tsur, J. Ullman, C. Clifton, S. Abiteboul, R. Motwani, S. Nestorov, and A. Rosenthal. Query Flocks: a Generalization of Association-Rule Mining. In *Proceedings of 1998 ACM SIGMOD, Seattle*, 1998.

[28] M.F. Worboys. GIS: A Computing Perspective. In *Taylor and Francis*, 1995.

[29] Y. Yasui and S.R. Lele. A Regression Method for Spatial Disease Rates: An Estimating Function Approach. *Journal of the American Statistical Association, 94:21-32*, 1997.