A Partial Join Approach for Mining Co-location Patterns [*]

Jin Soung Yoo
Department of Computer Science and Engineering
University of Minnesota
200 Union ST SE 4-192
Minneapolis, MN 55414
jyoo@cs.umn.edu

Shashi Shekhar
Department of Computer Science and Engineering
University of Minnesota
200 Union ST SE 4-192
Minneapolis, MN 55414
shekhar@cs.umn.edu

ABSTRACT

Spatial co-location patterns represent the subsets of events whose instances are frequently located together in geographic space. We identified the computational bottleneck in the execution time of a current co-location mining algorithm. A large fraction of the join-based co-location miner algorithm is devoted to computing joins to identify instances of candidate co-location patterns. We propose a novel *partial-join* approach for mining co-location patterns efficiently. It transactionizes continuous spatial data while keeping track of the spatial information not modeled by transactions. It uses a transaction-based *Apriori* algorithm as a building block and adopts the instance join method for residual instances not identified in transactions. We show that the algorithm is correct and complete in finding all co-location rules which have prevalence and conditional probability above the given thresholds. An experimental evaluation using synthetic datasets and a real dataset shows that our algorithm is computationally more efficient than the join-based algorithm.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Data mining, GIS and Spatial databases

General Terms

Algorithms

Keywords

Spatial data mining, Association rule, Co-location, Join

1. INTRODUCTION

A co-location represents a subset of spatial boolean events whose instances are often located in a neighborhood. Boolean spatial events describe the presence or absence of geographic object types at different locations in a two dimensional or three dimensional metric space, e.g., surface of the Earth. Examples of boolean spatial events include business types, mobile service request, disease, crime, climate, plant species, etc. Spatial co-location patterns may yield important insights for many applications. For example, a mobile service provider may be interested in service patterns frequently requested in a close location, e.g., 'today sales' and 'nearby stores'. The frequent neighboring request sets may be used for providing attractive location-sensitive advertisements, promotion, etc. Figure 1 shows the locations of different types of business in a downtown area of Minneapolis, Minnesota. We can notice two prevalent co-location patterns, i.e., {'auto dealers' , 'auto repair shops'} and {'department stores' , 'gift stores'}. Other application domains for co-locations are Earth science, environmental management, government services, public health, public safety, transportation, tourism, etc.

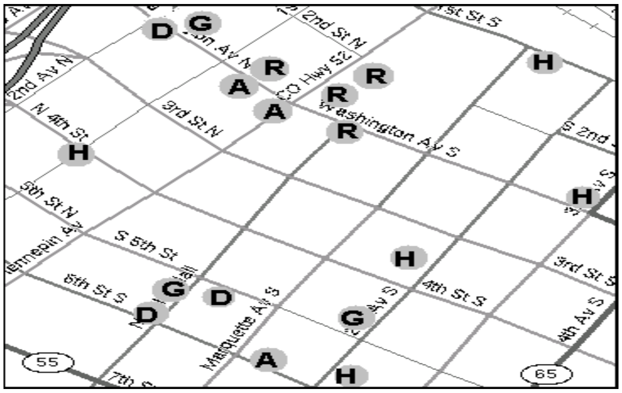
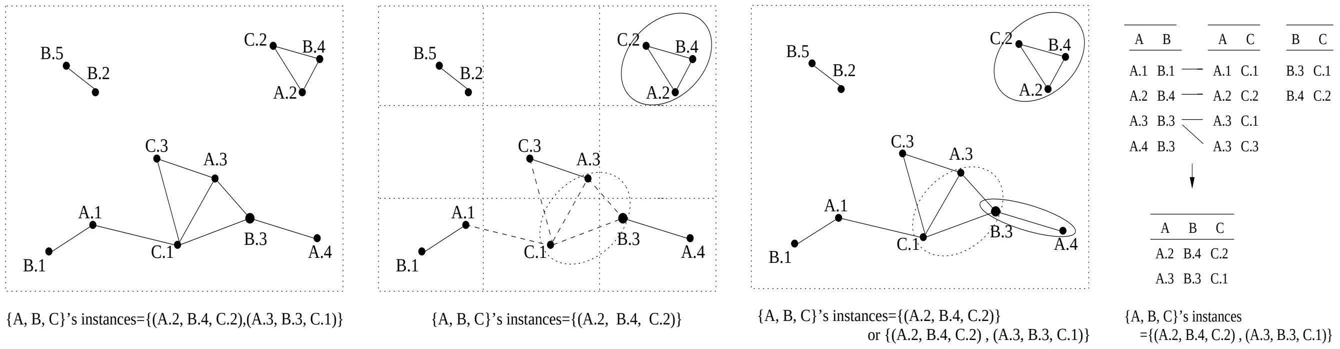


Figure 1: An example of co-location patterns found among different types of businesses in a city. A=Auto dealers, R=auto Repair shops, D=Department stores, G=Gift stores and H=Hotels.

[*] This work was partially supported by Digital Technology Center of University of Minnesota, NASA grant No. NCC 2 1231 and the Army High Performance Computing Research Center under the auspices of the Department of the Army, Army Research Laboratory cooperative agreement number DAAD19-01-2-0014, the content of which does not necessarily reflect the position or the policy of the government, and no official endorsement should be inferred

GIS'04, November 12–13, 2004, Washington, DC, USA.
Copyright 2004 ACM 1-58113-979-9/04/0011 ...$5.00.


B.5
B.2
C.2
B.4
A.2
C.3
A.3
A.1
B.1
C.1
B.3
A.4
{A, B, C}'s instances={(A.2, B.4, C.2),(A.3, B.3, C.1)}
{A, B, C}'s instances={(A.2, B.4, C.2)}
{A, B, C}'s instances={(A.2, B.4, C.2)}
or {(A.2, B.4, C.2) , (A.3, B.3, C.1)}
A B
A.1 B.1
A.2 B.4
A.3 B.3
A.4 B.3
A C
A.1 C.1
A.2 C.2
A.3 C.1
A.3 C.3
B C
B.3 C.1
B.4 C.2
A B C
A.2 B.4 C.2
A.3 B.3 C.1
{A, B, C}'s instances
={(A.2, B.4, C.2) , (A.3, B.3, C.1)}

(a) An example dataset (b) (c) (d)

Figure 2: Examples to illustrate different approaches to discover co-location patterns (b) An explicit transactionization of a spatial dataset can split instances of co-locations. (c) The non-overlapping grouping method can generate sets of different instances. (d) The instance join method generates complete instances but computation is expensive.

Co-location rule discovery is a process to identify co-location patterns from an instance dataset of spatial boolean events. It is not trivial to adopt association rule mining algorithms [1, 8, 13, 18] to mine co-location patterns since instances of spatial events are embedded in a continuous space and share a variety of spatial relationships. Reusing association rule algorithms may require transactionizing spatial datasets, which is challenging due to the risk of transaction boundaries splitting co-location pattern instances across distinct transactions. Figure 2 (a) shows an example spatial dataset with three spatial events, A, B, and C. Each instance is represented by its event type and unique instance id, e.g., A.1. Solid lines show neighbor relationships over event instances. For example, {A.2, B.4, C.2} and {A.3, B.3, C.1} are the instances of co-location {A, B, C} since their event instances are neighbors of each other. Figure 2 (b) shows the problem of explicit transactionization. Rectangular grids are used to produce transactions over the spatial dataset. As can be seen by the solid line circle, the only identified instance of co-location {A, B, C} is {A.2, B.4, C.2}. The instance {A.3, B.3, C.1} is missed due to the split caused by the transaction boundaries.

Related Work: In previous work on co-location pattern discovery, a few approaches have been developed to identify instances of candidate co-location patterns. One approach [12] groups neighboring instances arbitrarily with a non-overlapping instance grouping constraint. This disjoint grouping method may yield different instance sets by the order of grouping. For example, Figure 2 (c) illustrates different instance sets of co-location {A, B, C} by the order of grouping instances of size 2 co-location {A, B}. If an instance {A.4, B.3} is first grouped, the instance {A.3, B.3} is not identified since B.3 already belongs to instance {A.4, B.3} even if it is a neighborhood instance. Consequently, the instance {A.3, B.3, C.1} of co-location {A, B, C} is also not found.

Another approach [15] generates instances of candidate co-locations without any missing by using an instance join method. For example, in Figure 2 (d), the instances of co-location {A, B} and the instances of co-location {A, C} are joined and their neighbor relations are checked for generating instances of co-location {A, B, C}. {A.2, B.4, C.2} and {A.3, B.3, C.1} are correctly generated. The join-based algorithm may be useful in analyzing datasets of sparse instances. However, scaling the algorithm to substantially large dense spatial datasets is challenging due to the increasing number of co-location patterns and their instances. Other co-location mining work [17] presents a framework for extended spatial objects, e.g., polygons and line strings. It also uses an instance join method to identify nearby spatial objects.

This paper proposes a novel approach for efficient co-location pattern mining. We make the following contributions.

Our Contributions: First, we identified the computational bottleneck in the execution time of the join-based co-location mining algorithm [15]. A large fraction of the algorithm is devoted to computing joins to identify instances of candidate co-location patterns. Second, we propose a novel partial-join approach for mining co-location patterns efficiently. It transactionizes continuous spatial data while keeping track of the spatial information not modeled by transactions. This approach is based on an important observation that only event instances having at least one cut neighbor relation are related to co-location instances split over transactions. Third, we present an efficient co-location mining algorithm to concretize the partial-join approach. It uses a transaction-based *Apriori* algorithm [1] as a building block and adopts the instance join method [15] of the join-based co-location mining algorithm for generating residual co-location instances not identified by transactions. Fourth, we prove that the partial join algorithm is correct and complete in finding all co-location rules with prevalence and conditional probability above the given thresholds. Fifth, we provide an algebraic cost model to characterize the dominance zone of the performance between our partial-join algorithm and the join-based algorithm. Finally, we conducted experiments using a real dataset as well as synthetic datasets. The experimental evaluation shows that our algorithm is computationally more efficient than the full join-based mining algorithm.

The remainder of the paper is organized as follows. Section 2 presents an overview of basic concepts of co-location pattern mining. In Section 3, we present the partial join approach for efficient co-location mining. Section 4 describes the partial join co-location mining algorithm. The proofs of correctness and completeness of the algorithm, and an algebraic cost model are given in Section 5. Section 6 presents experimental evaluations. We give the conclusion and discuss future work in Section 7.

2. CO-LOCATION PATTERN MINING: BASIC CONCEPTS

This section describes the basic concepts for mining co-location patterns.

Given a set of boolean spatial events $E = \{e_1, \dots, e_k\}$, a set S of their instances $\{i_1, \dots, i_n\}$, and a reflexive and symmetric neighbor relation R over S, a **co-location** C is a subset of boolean spatial events, i.e., $C \subseteq E$ whose instances $I \subseteq S$ form a clique [3] using neighbor relation R. For simplicity, we use a metric-based neighbor relation R, i.e., neighbor(i_1, i_2) between event instances i_1 and i_2 defined by Euclidean_distance(i_1, i_2) $\leq$ a user-specified threshold is used as a neighbor relation R.

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

A neighborhood instance I of a co-location C is a **row instance** (simply, instance) of C if I contains instances of all events in C and no proper subset of I does so. For example, in Figure 2 (d), {A.1, B.1} is a row instance of co-location {A, B}. {A.3, C.1, C.3} is a neighborhood in Figure 2 (a) but it is not a row instance of co-location {A, C} because its subset {A.3, C.1} contains instances of all events in {A, C}. The **table instance** of a co-location C is the collection of all row instances of C. For example, the table instance of {B, C} in Figure 2 (d) has two row instances, {B.3, C.1} and {B.4, C.2}.

The **conditional probability**, $Pr(C_1|C_2)$, of a co-location rule $C_1 \rightarrow C_2$ is the probability of finding an instance of C_2 in the neighborhood of an instance of C_1. Formally, it is estimated as $\frac{|\pi_{C_1}(table\, instance\, of\, C_1 \cup C_2)|}{|table\, instance\, of\, C_1|}$, where π is a projection operation with duplication elimination.

The **participation index**, $Pi(C)$ is used as a co-location prevalence measure. The participation index of a co-location $C = \{e_1, \dots, e_k\}$ is defined as $\min_{e_i \in C}\{Pr(C, e_i)\}$, where $Pr(C, e_i)$ is the **participation ratio** for event type e_i in a co-location C. $Pr(C, e_i)$ is the fraction of instances of e_i which participate in any instance of co-location C, $\frac{|\pi_{e_i}(table\, instance\, of\, C)|}{|table\, instance\, of\, e_i|}$, where π is a projection operation with duplication elimination. For example, in Figure 2 (a), the total number of instances of event type A is 4 and the total number of instances of event type C is 3. From Figure 2 (d), the participation index of co-location c={A, C} is $min\{Pr$(c, A), Pr(c,C)} = 3/4 because Pr(c, A) is 3/4 and Pr(c,C) is 3/3. A high participation index value indicates that the spatial events in a co-location pattern likely show up together.

LEMMA 1. *The participation ratio and the participation index are monotonically non increasing with the size of the co-location increasing.*

PROOF. Please refer to [15] for the proof. □

Lemma 1 ensures that the participation index can be used to effectively prune the search space of co-location pattern mining.

3. PARTIAL JOIN APPROACH FOR CO-LOCATION PATTERN MINING

This section defines our partial join approach for efficient co-location pattern mining.

3.1 Problem Definition

We formalize the co-location mining problem as follows:

Given:

1) A set of k spatial event types $E = \{e_1, \dots, e_k\}$ and a set of their instances $S = \{i_1, \dots, i_n\}$, each $i \in S$ is a vector < instance id, spatial event type, location >, where location $\in$ a spatial framework

2) A symmetric and reflexive neighbor relation R over locations

3) A minimal prevalence threshold (*min_prev*) and a minimal conditional probability threshold (*min_cond_prob*)

Find:

Find a correct and complete set of co-location rules with participation index > *min_prev* and conditional probability > *min_cond_prob*.

Objective:

Minimize computation cost.

Constraints:

1) R is a distance metric based neighbor relation.

2) Ignore edge effects in R.

3) Correct and complete in finding all co-location rules satisfying given thresholds.

4) Spatial dataset is a point dataset.

3.2 Partial Join Approach

The basic idea of the partial join approach is to reduce the number of instance joins for identifying instances of candidate co-locations by transactionizing a spatial dataset under a neighbor relationship and tracing only residual neighborhood instances cut apart via the transactions. The key component of our approach is how we identify instances of co-locations split across explicit transactions. It is based on an observation that only event instances having at least one cut neighbor relationship are related to the neighborhood instances split over transactions. To formalize this idea, we provide a set of definitions of key terms related to the partial join approach.

DEFINITION 1. *A* **neighborhood transaction***(simply, transaction) is a set of instances $T \subseteq S$ that forms a clique using a neighbor relation R. A spatial dataset S is partitioned to a set of disjoint transactions $\{T_1, \dots, T_n\}$ where $T_i \cap T_j = \emptyset$, $i \neq j$ and $\cup(T_1, \dots, T_n) = S$.*

We assume a spatial dataset S can be partitioned to a set of distinct transactions, i.e., each event instance $i \in S$ belongs to one transaction. For example, Figure 4 shows a set of transactions on the same example spatial dataset of Figure 2 (a). The dashed circle represents a neighborhood region centered at an arbitrary location on a spatial framework. The instances within the dashed circle are neighbors of each other and thus forms a transaction. For example, B.2 and B.5 form a transaction. A spatial dataset can be differently transactionized according to the partitioning method used. Thus the transactions generated using rectangular grids in Figure 2 (b) are a little different from the transactions illustrated in Figure 4. For example, in Figure 4, {A.3, C.1, C.3} forms a single transaction. By contrast, in Figure 2 (b), it is divided into two transactions, {A.3, C.3} and {C.1}. We will examine the effect of different transactionization methods in future work.

Definition 2. *A row instance I of a co-location C is an* **intraX row instance** *(simply, intraX instance) of C if all instances $i \in I$ belong to a common transaction T. The* **intraX table instance** *of C is the collection of all intraX row instances of C.*

For example, in Figure 4, {A.3, C.1} is an intraX instance of co-location {A, C} but {A.1, C.1} is not since its event instances A.1 and C.1 are members of different transactions. The intraX table instance of {A, C} consists of {A.3, C.1}, {A.3, C.3} and {A.2, C.2}.

Definition 3. *A neighbor relation $r \in R$ between two event instances, $i_1, i_2 \in S, i_1 \neq i_2$ is called a* **cut neighbor relation** *if i_1 and i_2 are neighbors of each other but belong to distinct transactions.*

Figure 4 presents cut neighbor relations as dotted lines. {A.1, C.1}, {A.3, B.3} and {B.3, C.1} has cut neighbor relations.

Definition 4. *A row instance I of a co-location C is an* **interX row instance** *(simply, interX instance) of C if all instances $i \in I$ have at least one cut neighbor relation. The* **interX table instance** *of C is the collection of all interX row instances of C.*

For example, in Figure 4, {A.3, B.3} is an interX instance of co-location {A, B} because A.3 has a cut neighbor relation with B.3 and B.3 also has cut neighbor relations with A.3 and with C.1. Note {A.3, C.1} is an interX instance as well as an intraX instance of {A, C}. InterX table instance of {A, C} has two interX instances {A.1, C.1} and {A.3, C.1}.

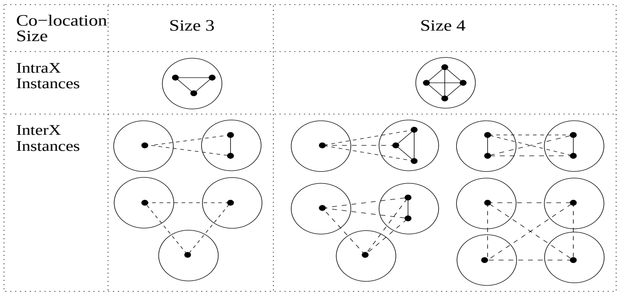
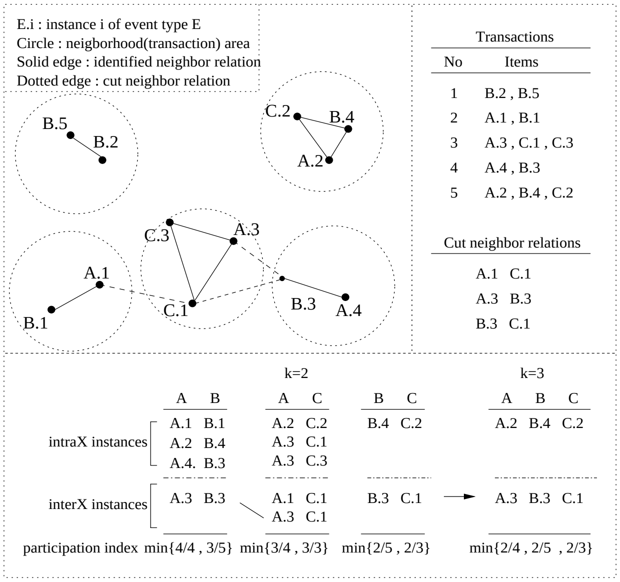


Figure 3: The cases of possible instances of size 3 and of size 4 co-locations over transactions

Figure 3 illustrates the possible instances of size 3 co-location and of size 4 co-location located over neighborhood transactions. Black dots signify event instances, circles are transactions, and lines show neighbor relations between two event instances. Especially, dotted lines signify cut neighbor relations. There are two types of instances of co-locations. One is all event instances of a co-location instance belong to a single transaction. The other is the event instances are distributed across two or more transactions. The former is the case of an intraX instance and the latter is an interX instance. We can notify all event instances of interX instances are related to at least one cut neighbor relation(dotted lines).


E.i : instance i of event type E
Circle : neigborhood(transaction) area
Solid edge : identified neighbor relation
Dotted edge : cut neighbor relation
B.5
B.2
C.2
B.4
A.2
C.3
A.3
A.1
C.1
B.1
B.3
A.4

Transactions	
No	Items
1	B.2 , B.5
2	A.1 , B.1
3	A.3 , C.1 , C.3
4	A.4 , B.3
5	A.2 , B.4 , C.2

Cut neighbor relations	
A.1	C.1
A.3	B.3
B.3	C.1

	k=2						k=3		
	A	B	A	C	B	C	A	B	C
intraX instances	A.1	B.1	A.2	C.2	B.4	C.2	A.2	B.4	C.2
	A.2	B.4	A.3	C.1					
	A.4.	B.3	A.3	C.3					
interX instances	A.3	B.3	A.1	C.1	B.3	C.1	A.3	B.3	C.1
			A.3	C.1					
participation index	min{4/4 , 3/5}		min{3/4 , 3/3}		min{2/5 , 2/3}		min{2/4 , 2/5 , 2/3}		

Figure 4: An illustration of the partial join co-location mining algorithm

Lemma 2. *For a co-location C, the table instance of C is the union of intraX table instance of C and interX table instance of C.*

Proof. The table instance of a co-location C is the collection of all (row) instances of C. First, we will show any instance, $I = \{i_1, \dots, i_n\}$ of C is an intraX instance of C or an interX instances of C. Since I forms a clique using a neighbor relation, all event instances of I can be included in a single neighborhood transaction according to definition 1. I becomes an intraX instance. By contrast, if all event instances of I are not in a single transaction, each member should have at least one cut neighborhood relation with the other members in different transactions due to their clique relation. Thus, I becomes an interX instance. Second, all instances of intraX table instance and interX table instance of C are row instances whose event instances form a clique according to definition 2 and definition 4 . □

4. PARTIAL JOIN CO-LOCATION MINING ALGORITHM

This section describes the partial join co-location mining algorithm. A transaction-based *Apriori* algorithm [1] is used as a building block to identify all intraX instances of co-locations. InterX instances are generated using *generalized apriori_gen* function [15] of the join-based co-location mining algorithm. This approach is expected to provide a framework for efficient co-location mining since all instances in the transaction are neighbors of each other and no spatial operation and combinatorial operation, i.e., join, is required to find instances of candidate co-locations within a transaction, i.e., intraX instances. The computation cost of instance join operations for generating only interX instances not identified in the transactions is relatively cheaper than one for finding all instances of co-locations. The partial-join mining algorithm for co-location patterns is described as follows.

```
Inputs
    $E$:a set of boolean spatial event types
    $S$:a set of instances
        <event type, event instance id, location>
    $R$:a spatial neighbor relation
    $min\_prev$:prevalence value threshold
    $min\_cond\_prob$:conditional probability threshold
Output
    A set of all prevalent co-location rules with
    participation index greater than $min\_prev$
    and conditional probability greater than
    $min\_cond\_prob$
Variables
    $k$:co-location size
    $T$:a set of transactions
    $C_k$:a set of size $k$ candidate co-locations
    $P_k$:a set of size $k$ prevalent co-locations
    $R_k$:a set of size $k$ co-location rules
    $IntraX_k$:intraX table instances of $C_k$
    $InterX_k$:interX table instances of $C_k$, $P_k$
Method
($T$, $InterX_2$)=transactionize($S$, $R$);
$k = 1$; $C_1 = E$; $P_1 = E$;
while (not empty $P_k$) do {
  $C_{k+1}$=gen_candidate_co-location($P_k$);
  for all transaction $t \in T$
    $IntraX_{k+1}$=gather_intraX_instances($C_{k+1}, t$);
  if $k \geq 2$
    $InterX_{k+1}$=gen_interX_intances($C_{k+1}, InterX_k, R$);
  $P_{k+1}$=select_prevalent_co-location
          ($C_{k+1}, IntraX_{k+1} \bigcup InterX_{k+1}, min\_prev$);
  $R_{k+1}$=gen_co-location_rule($P_{k+1}, min\_cond\_prob$);
  $k = k + 1$;
}
return $\bigcup(R_2, \ldots, R_{k+1})$;
```

Algorithm 1: Partial join co-location algorithm

Transactionization of a spatial dataset : Given a spatial dataset and a neighbor relation, the spatial dataset is partitioned for generating neighborhood transactions. There are several partitioning methods adopted for neighborhood transactions, e.g., grids [14], maximal cliques[3], max-clique agglomerative clustering [20], min cut partitioning [6] etc. The ideal case is a method to generate a set of maximal cliques with minimizing the number of edges cut by partitions. In the case of a simple grid partitioning, rectangular grids of a proximity neighborhood size $d \times d$, where d is a neighbor distance metric, are posed on a spatial framework, and event instances in each cell are gathered for a transaction. Cut neighbor relations can be detected by examining all pairs (i_1, i_2) of instances in neighboring cells, i.e., $(i_1, i_2) \in R$ and $i_1.trans\_no \neq i_2.trans\_no$, where R is a neighbor relation. It can be implemented using geometric approaches, e.g., plane sweep [2], space partitioning [9], tree matching [10]. Size 2 interX instances are generated from all pairs(i_1, i_2) of instances having cut neighbor relations in each transaction, i.e., $i_1 \in B$, $i_2 \in B$ and $i_1.trans\_no = i_2.trans\_no$, where B is a set of event instances having cut neighbor relations , as well as cut neighborhood instances.

Generation of candidate co-locations : We use the *apriori_gen* [1] for generating candidate co-location sets. Size $k + 1$ candidate co-locations are generated from size k prevalent co-locations. The anti-monotonic property of the participation index makes event level pruning feasible.

Scanning transactions and gathering intraX instances : In each iteration step, the transactions are scanned and the intraX instances of candidate co-locations are enumerated. This step is similar to the apriori algorithm. However, notice that the transactions of a spatial event dataset differ from the transactions of a market basket dataset. The traditional market basket data transaction has only boolean item types, i.e., an item is present in a transaction or not. By contrast, each item of our neighborhood transaction consists of an event type and its instance id as described in Figure 4. One event type can have several instances in a transaction. To reuse an efficient trie data structure [4, 7] in determining instances of candidate co-locations in a transaction, we convert several items of same event type with different instance ids to one event type item having a bitmap structure [5] in which corresponding instance id bits are set. The converted transactions are searched for gathering intraX instances of co-locations. Figure 4 shows a conceptual set of intraX table instances. Actually, all instances are enumulated in the trie structure of itemsets using bitmaps.

Generation of interX table instances : The interX table instance of C_{k+1}, $k \geq 2$ are generated from interX table instance of C_k using the *generalized apriori_gen* function [15]. The SQL-like syntax is described below.

```
forall co-location $c_{k+1} \in C_{k+1}$
  insert into $c_{k+1}$.interX_table_instance
  select $p$.instance$_1$, $p$.instance$_2$, ..., $p$.instance$_k$
       , $q$.instance$_k$
  from $c_k$.interX_table_instance$_1$ $p$
       , $c_k$.interX_table_instance$_2$ $q$
  where ($p$.instance$_1$, ..., $p$.instance$_{k-1}$)
        = ($q$.instance$_1$, ..., $q$.instance$_{k-1}$)
     and ($p$.instance$_k$, $q$.instance$_k$) $\in R$;
end;
```

In Figure 4, an interX table instance of {A, B} having {A.3, B.3} and an interX table instance of {A, C} having {A.1, C.1} and {A.3, C.1} are joined to produce interX table instance of {A, B, C}.

Selection of Prevalent Co-locations: The participation index of co-location C_{k+1} is calculated from the union of intraX table instance(C_{k+1}) and interX table instance(C_{k+1}). Candidate co-locations are pruned using a given prevalence threshold, *min_prev*. In Figure 4, co-location {B, C} has two instances, i.e., one is an intraX instance, {B.4, C.2} and the other is an interX instance {B.3, C.1}. The participation index of co-location {B, C} is $min\{2/5,\ 2/3\} = 2/5$.

If *min_prev* is given as 1/2, the candidate co-location {B, C} is pruned because its prevalence measure is less than 1/2.

Generation of Co-location Rules: This step generates all co-location rules with high conditional probability above a given *min_cond_prob*.

5. ANALYSIS OF THE PARTIAL JOIN CO-LOCATION MINING ALGORITHM

In this section, we analyze the partial join co-location mining algorithm for completeness, correctness and computational complexity. Completeness implies that no co-location rule satisfying given prevalence and conditional probability thresholds is missed. Correctness means that the participation index values and conditional probability of generated co-location rules meet the user specified threshold.

5.1 Completeness and Correctness

Lemma 3. *The partial join co-location mining algorithm is correct.*

Proof. The partial join co-location mining algorithm is correct if co-location patterns produced by algorithm 1 meets the thresholds of prevalence value and conditional probability. First, we will show that intraX instances and interX instances are correct in the neighbor relation. Step 1 in algorithm 1 generates neighborhood transactions according to definition 1. Thus the intraX instances gathered in step 6 are correct in the neighbor relation. The interX instances generated in step 8 are proved by the correctness of *generalized apriori_gen* algorithm [15]. That is, all instances of a generated interX instance are neighbor of each other. Second, step 9 ensures that only prevalent co-location sets are selected. Thus step 11 returns co-location rules above given thresholds correctly. □

Lemma 4. *The partial join co-location mining algorithm is complete.*

Proof. We prove if a co-location is prevalent, it is found by algorithm 1. First, the monotonicity of the participation index in lemma 1 proves the completeness of the event level pruning of candidate co-locations using *apriori_gen* in step 4. Second, we will show that the intraX table instances and the interX table instances generated from algorithm 1 are complete, which will imply that all instances of co-locations are complete according to lemma 2. All intraX table instances are completely found by the apriori algorithm in step 6. Size 2 interX table instances generated from step 1 are a superset of all neighboring instances necessary to generate size $k+1$, $k \geq 2$ interX instances. In step 8, the completeness of the instance join method to generate interX instances is the same as that of *generalized apriori_gen* [15]. In step 11, enumeration of the subsets of each of the prevalence co-locations ensures that no spatial co-location rules satisfying given prevalence and conditional probabilities are missed. □

5.2 Computational Complexity Analysis

This section compares the computational cost of the join-based co-location mining algorithm and the partial join algorithm. Let $T_{jb}(k+1)$ and $T_{pj}(k+1)$ represent the costs of iteration k of the join-based algorithm and the partial join algorithm respectively.

$$\begin{aligned} T_{jb}(k+1) &= T_{gen\_candi}(P_k) \\ &\quad + T_{gen\_inst}(table\_insts\,of\,P_k) + T_{prune}(C_{k+1}) \\ &\approx T_{gen\_inst}(table\_insts\,of\,P_k) \end{aligned}$$

$$\begin{aligned} T_{pj}(k+1) &= T_{gen\_candi}(P_k) + T_{gath\_intraX\_inst}(transactions) \\ &\quad + T_{gen\_interX\_inst}(interX\_table\_insts\,of\,P_k) + T_{prune}(C_{k+1}) \\ &\approx T_{gen\_interX\_inst}(interX\_table\_insts\,of\,P_k) \end{aligned}$$

In the above equations, $T_{gen\_candi}(P_k)$ represents the cost of generating size $k+1$ candidate co-location with the prevalent size k co-locations. $T_{gen\_inst}(table\_insts\,of\,P_k)$ represents the cost of generating table instances of size $k+1$ candidate co-locations with size k table instances. $T_{gath\_intraX\_inst}(transactions)$ is the cost of scanning transactions and gathering the instances of the size $k+1$ candidate co-locations. $T_{gen\_interX\_inst}(interX\_table\_inst\,of\,P_k)$ is the cost of generating interX table instances of the size $k+1$ candidate co-locations with size k interX table instances. $T_{prune}(C_{k+1})$ represents the cost for pruning non prevalent size $k+1$ co-locations.

The bulk of time is consumed in generating instances. We assume that the cost of gathering intraX instances from transactions is relatively cheaper than instance join cost, and that the other factors, $T_{gen\_candi}(P_k)$ and $T_{prune}(C_{k+1})$ are illegible. Thus the computational ratio of the partial join algorithm over the join-based algorithm can be simplified as

$$\frac{T_{pj}(k+1)}{T_{jb}(k+1)} \approx \frac{T_{gen\_interX\_inst}(interX\_table\_insts\,of\,P_k)}{T_{gen\_inst}(table\_insts\,of\,P_k)}$$

The computational ratio is affected by the size of interX table instances and the size of table instances of co-location P_k. The dominance factors affecting the number of interX instances and the number of total instances can be the number of cut neighbor relations and the data density of the neighborhood area. When the number of cut neighbor relations is fixed and the data density in a neighborhood area grows, the size of table instances increases rapidly and the cost to generate the table instances is much greater than the cost to generate interX table instances. By contrast, as the number of cut neighbor relations increases, the size of interX table instances increases. Thus the average cost to generate interX table instances grows. When all instances have cut neighbor relations, they are involved in interX table instances thus the cost to generate the interX table instances is similar to the cost to generate table instances in the join-based algorithm. In our experiments, as described in the next section, we use the data density in neighborhood area and the ratio of cut neighbor relations as key parameters to evaluate the algorithms. We can expect that the partial join approach is likely more efficient than the join-based method when the locations of spatial events are clustered in neighborhood areas and the number of cut neighbor relations is smaller.

6. EXPERIMENTAL EVALUATION

We evaluated the performance of the partial join algorithm with the join-based approach using synthetic and real datasets. In Subsection 6.1, we describe an overall experimental design and a synthetic data generator. In Subsection 6.2, we evaluate the computational efficiency gained

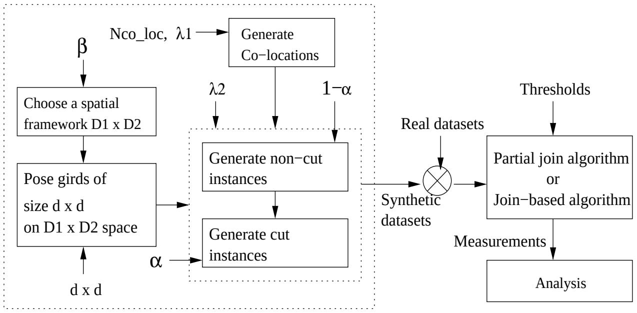

Nco_loc, λ1
Generate Co−locations
β
Choose a spatial framework D1 x D2
λ2
1−α
Generate non−cut instances
Generate cut instances
Pose girds of size d x d on D1 x D2 space
α
d x d
Real datasets
Synthetic datasets
Thresholds
Partial join algorithm or Join−based algorithm
Measurements
Analysis

Figure 5: Experimental Design

from our partial join co-location algorithm with synthetic datasets by studying the parameters that affect performance. Subsection 6.3 compares the performance of the algorithms using a real dataset.

6.1 Experiment Design

Figure 5 shows an overall experiment layout. Synthetic datasets were generated using a methodology similar to the methodology used to evaluate the join-based algorithm [15]. We added some parameters and procedures in it to generate transactionized instances and cut neighbor relations. The synthetic data generator allows better controls in studying the effects of interesting parameters. First we describe the layout of an overall spatial framework. For simple transactionization of a spatial dataset, we posed grids of neighborhood size $d \times d$ on a rectangle spatial framework of size $D_1 \times D_2$. Each grid cell is implicitly divided into two parts, a core area and an overlapping area. The core area is an area in which event instances have neighbor relationships with only instances in its grid cell. By contrasts, instances in the overlapping area are also under neighbor relations with instances in its neighboring cells. This area was used for generating cut neighbor relations.

The synthetic spatial datasets were generated as follows. Given a number of base co-location patterns, Nco_loc, the size of each co-location n_1 was picked from a Poisson distribution with mean λ_1. We assigned randomly chosen sets of event types to the co-location patterns. The number of base instances of each co-location n_2 was chosen from another Poisson distribution with mean λ_2. Our data generator is also controlled by two other parameters, cut instance ratio α and spatial framework size β. The cut instance ratio was used for controlling the number of cut neighbor relations in the experiment. $(1-\alpha) * n_2$ instances were generated in the core area of a randomly chosen cell. $\alpha * n_2$ instances were generated over its overlapping area and the overlapping areas of its neighboring cells. For simply controlling the data density value under datasets of the same size, we changed the size of the overall spatial framework. To increase the density value, we used a smaller spatial framework but the same neighborhood size $d \times d$.

The partial join co-location algorithm and the join-based co-location algorithm were executed using generated spatial datasets and a real set of climate data from NASA. The performance of the two algorithms was evaluated by execution time. The average co-location size and the average number of instances of co-locations of the generated datasets are likely different from the initial parameter values after generating cut instances and also according to the size of choosen spatial framework for controlling the data density. We will address the effect of these parameters and noise data on performance in future work. All the experiments were performed on a Sun SunBlade 1500 with 1.0 GB main memory and 177MHz CPU.

6.2 Performance Study

The experiment was conducted using detailed simulations to answer the following questions :
1. How does the ratio of cut neighbor relations over total neighbor relations affect the performance ?
2. How does data density in the neighborhood area affect the performance ?
3. How do the algorithms behave with different prevalence thresholds ?

The common parameter values used in these experiments were as follows: the neighborhood size to define a co-location, $d \times d$, is 10×10, the number of base co-locations, Nco_loc, is 20, the average size of co-location patterns, λ_1, is 4 and the average size of co-location instances, λ_2, is 50.

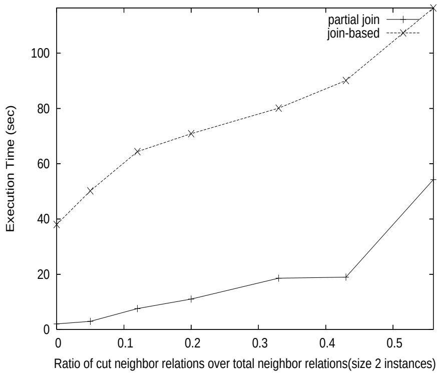


Figure 6: Effect of ratio of cut neighbor relations over total neighbor relations

Effect of ratio of cut neighbor relations : The effect of performance by the ratio of cut neighbor relations over total neighbor relations was evaluated with synthetic datasets generated using the above common parameters and different cut instance ratios, i.e., 0, 0.1, 0.2, 0.3, etc. The size of the overall spatial framework was fixed to 400×400. The prevalence threshold was set to 0.2.

Figure 6 shows the execution time of both algorithms, the partial join and the join-based, over cut neighbor relation ratios. The ratio of cut neighbor relations over total neighbor relations was controlled by the cut instance ratio in the experiment. The overall execution time increased with increases in the ratio. The reason is, that as the ratio of cut relations becomes larger, the size of interX table instances increases. This causes the number of instances involved in the join operation to grow and the execution time to increase. The join-based algorithm also shows an increase in its execution time. This happens because the number of instances in the overlapping area increases and the possibility of neighbor relations with instances in the nearby cells increases, thus generating many neighborhood instances. The average size of table instances also increases.

Table 1: A comparison of size 2 instances

Data density in the neighborhood area	Number of size 2 interX instances of partial join	Number of size 2 instances of join-based
0.07	2,429	18,450
0.09	2,892	24,696
0.10	3,426	30,233
0.12	3,496	31,099
0.13	4,268	39,583

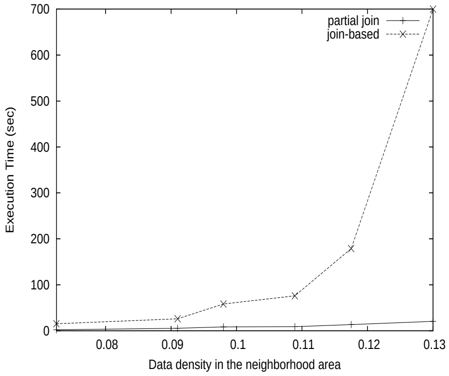


Figure 7: Effect of data density on neighborhood area

The performance difference between the two algorithms decreases with increases in the number of cut neighborhoods. When all event instances were related to cut neighbor relations, the two algorithms showed similar execution time.

Effect of data density in the neighborhood : The effect of data density in the neighborhood area was evaluated with spatial datasets generated using the above common parameters and spatial frameworks of different size β, i.e., 500×500, 400×400, 360×360, etc., to control the data density on the neighborhood. The cut instance ratio α was fixed to 0.1 and the prevalence measure was set to 0.2. The density value was calculated from the generated dataset. It is the ratio of the average number of instances in a neighborhood area over the size of the square neighborhood area, 10×10. The increase of data density in this experiment mainly affects to data density in the core area since the cut instance ratio is fixed.

Figure 7 illustrates the performance gain by the partial join algorithm. As the density increases, the execution time of the join-based algorithm is dramatically increased. By contrast, the partial join algorithm shows little effect from the size of data density in the neighborhood area. A small-scale increase of data in the the neighborhood area does not much affect the transaction-based algorithm while the join-based method shows great sensitivity to even a small increase of the data density. Table 1 shows a comparison between the number of size 2 interX instances generated by the partial join algorithm and the number of size 2 instances of the join-based algorithm. These instances are involved in instance join operations for generating size 3 instances. As can be seen, the partial join approach had much fewer instances than the join-method in this experiment.

Effect of prevalence threshold : The performance effect as the prevalence threshold increases is given in Figure 8. The experiment was conducted with the above common parameters, a 400×400 spatial framework and a 0.3 cut instance ratio. The partial join approach showed much better performance than the join-based approach when the threshold values were low. However, the gap dramatically decreased with increases in the threshold value. The reason is the decrease in the number of joins of instances due to the efficient pruning of the event level search space.

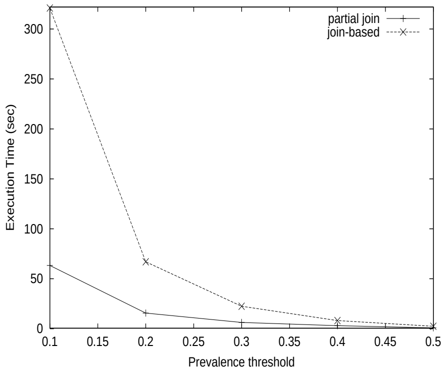


Figure 8: Effect of prevalence threshold

6.3 Experiment on a Real Dataset

We evaluated the partial join algorithm and the join-based algorithm using a NASA climate dataset of the U.S. region. All events were extracted at the threshold 1.5 using Z score transformation [16]. The number of event types was 18. The total number of event instances was 15,515. When the neighborhood distance threshold was 4, the total number of size 2 neighborhood instances was 390,392 and the number of size 2 cut neighborhood instance(size 2 interX instances) was 314,078. When the prevalence threshold was 0.1, the maximum size of co-locations was 5. Figure 9 presents the execution time of the two algorithms as a function of the prevalence threshold. The partial join method shows relatively better performance.

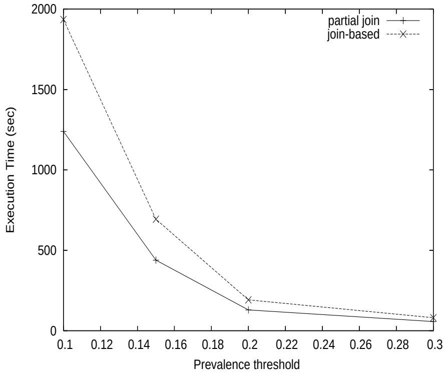


Figure 9: A comparison using a real dataset

7. CONCLUSION AND FUTURE WORK

In this paper, we identified the limitations of the current co-location mining algorithm and proposed a novel partial-join approach for mining complete and correct co-location patterns. This approach transactionizes continuous spatial data while keeping track of the spatial information not modeled by transactions. To concretize this approach, we proposed an efficient partial join co-location algorithm to adopt the instance join method on the framework of the *Aprioir* algorithm. We provided an algebraic cost model to characterize the dominance zone of the performance between the partial-join approach and the join-based method. The performance study showed that our approach is computationally more efficient and is especially robust in data density on the neighborhood.

In future work, first, we plan to develop an alternative efficient join algorithm for generating instances of co-locations without including duplicate instances in intraX table instances and interX table instances. This approach will further reduce the number of instance joins. The algorithm will be robust in any dataset, e.g., dense datasets with many cut neighborhoods. Second, we used a regular grid based transactionziation. We plan to examine different transactionization methods, e.g., maximal cliques[3], max-clique agglomerative clustering [20], min cut partitioning [6] etc. Third, recent work [19] on the co-location mining presents a general method to find the maximal patterns of reference feature centric co-locations and clique co-locations considering memory constraints. It uses techniques of spatial join algorithms, e.g., partition-based spatial join [9] , multiway spatial join [11]. We plan to compare our approach to the spatial join-based method. Finally, although current co-location patterns are defined over spatial features, data in many applications include spatio-temporal features. Thus we also plan to explore a co-location mining method for spatio-temporal datasets.

8. ACKNOWLEDGMENTS

We thank the spatial database group at the University of Minnesota for their support. We would also like to express our thanks to Kim Koffolt for her timely and detailed feedback to help improve the readability of this paper.

9. REFERENCES

[1] R. Agarwal and R. Srikant. Fast algorithms for Mining association rules. In *Proc. of the 20th VLDB*, 1994.

[2] L. Arge, O. Procopiuc, S. Ramaswamy, T. Suel, and J. Vitter. Scalable Sweeping-Based Spatial Join. In *Proc. of the Int'l Conference on Very Large Databases*, 1998.

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

[4] C. Borgelt. Efficient implementations of apriori and eclat. In *Workshop of Frequent Item Set Mining Implementations FIMI*, 2003.

[5] R. Elmasri and S. Navathe. *Fundamentals of Database Systems.* Addison Wesley Higher Education, ISBN 0-201-74153-9, 2002.

[6] G.Karypis and V. Kumar. Multilevel k-way Partitioning Scheme for Irregular Graphs . In *the Journal of Parallel and Distributed Computing*, 1995.

[7] B. Goethals. Frequent pattern mining implementations. http://www.cs.helsinki.fi/u/goethals/software/index.html.

[8] J. Hipp, U. Güntzer, and G. Nakjaeizadeh. Algorithms for Association Rule Mining - A General Survey and Comparison. *ACM SIGKDD Explorations*, 2(1), 2000.

[9] D. J. D. J. M. Patel. Partition Based Spatial-Merge Join. In *Proc. of the ACM SIGMOD Conference on Management of Data, pp.259-270, Montreal, Canada*, June 1996.

[10] S. T. Leutenegger and M. A. Lopez. The Effect of Buffering on the Performance of R-Trees. In *Proc. of the International Conference on Data Engineering, pp 164-171*, 1998.

[11] N. Mamoulis and D. Papadias. Multiway spatial joins. In *ACM Transactions on Database System*, 2001.

[12] Y. Morimoto. Mining Frequent Neighboring Class Sets in Spatial Databases. In *Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2001.

[13] A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association rules in large databases. In *Proc. of the 21st VLDB Conference*, 1995.

[14] S. Shekhar and S. Chawla. *Spatial Databases: A Tour.* Prentice Hall, ISBN 0130174807, 2003.

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

[16] P. Tan, M. Steinbach, V. Kumar, C. Potter, S. Klooster, and A. Torregrosa. Finding spatio-temporal patterns in earth science data. In *Proc. of KDD Workshop on Temporal Data Mining*, 2001.

[17] H. Xiong, S. Shekhar, Y. Huang, V. Kumar, X. Ma, and J. S. Yoo. A Framework for Discovering Co-location Patterns in Data Sets with Extended Spatial Objects. In *Proc. 2004 SIAM International Conference on Data Mining (SDM)*, 2004.

[18] M. J. Zaki and S. Parthasarathy. New algorithms for fast discovery of association rules. In *Proc. of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1997.

[19] X. Zhang, N. Mamoulis, D. Cheung, and Y. Shou. Fast Mining of Spatial Collocations. In *Proc. of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2004.

[20] Y. Zhao and G. Karypis. Evaluation of Hierarchical Clustering Algorithms for Document Datasets. In *Proc. of ACM Conference on Information and Knowledge Management(CIKM)*, 2002.