A Joinless Approach for Mining Spatial Colocation Patterns

Jin Soung Yoo, *Student Member*, *IEEE*, and Shashi Shekhar, *Fellow*, *IEEE*

Abstract—Spatial colocations represent the subsets of features which are frequently located together in geographic space. Colocation pattern discovery presents challenges since spatial objects are embedded in a continuous space, whereas classical data is often discrete. A large fraction of the computation time is devoted to identifying the instances of colocation patterns. We propose a novel joinless approach for efficient colocation pattern mining. The joinless colocation mining algorithm uses an instance-lookup scheme instead of an expensive spatial or instance join operation for identifying colocation instances. We prove the joinless algorithm is correct and complete in finding colocation rules. We also describe a partial join approach for spatial data which are clustered in neighborhood areas. We provide the algebraic cost models to characterize the performance dominance zones of the joinless method and the partial join method with a current join-based colocation mining method, and compare their computational complexities. In the experimental evaluation, using synthetic and real-world data sets, our methods performed more efficiently than the join-based method and show more scalability in dense data.

Index Terms—Spatial data mining, association rule, colocation pattern, spatial neighbor relationship.

1 Introduction

The explosive growth of spatial data and widespread use of spatial databases emphasize the need for the automated discovery of spatial knowledge. Spatial data mining [1], [2] is the process of discovering interesting and previously unknown, but potentially useful patterns from large spatial databases. Extracting interesting patterns from spatial data sets is more difficult than extracting the corresponding patterns from traditional numeric and categorical data due to the complexity of spatial data types, spatial relationships, and spatial autocorrelation [3].

A spatial colocation represents a subset of spatial features whose instances are frequently located together in spatial neighborhoods. For example, epidemiologists have found that West Nile disease and stagnant water sources are frequently colocated. The colocation rule, i.e., stagnant water source → West Nile disease, predicts the presence of West Nile disease in areas with stagnant water sources. Spatial colocation patterns may yield important insights for many applications. For example, a mobile service provider may be interested in service patterns frequently requested by geographically neighboring users. The frequent neighboring request sets may be used for providing attractive location-sensitive advertisements, recommendations, etc. In another example, ecologists are interested in animal behaviors that frequently co-occur with their nearby environmental or social factors, e.g., food abundance and alpha attack [4]. Other application domains include Earth science, biology, public health, transportation, etc.

• *The authors are with the Computer Science Department, University of Minnesota, 4-192, EE/CS Bldg., 200 Union Street SE, Minneapolis, MN 55455. E-mail: {jyoo, shekhar}@cs.umn.edu.*

Manuscript received 1 Sept. 2005; revised 8 Feb. 2006; accepted 17 Apr. 2006; published online 18 Aug. 2006.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-0345-0905.

Colocation rule discovery is a process to identify colocation patterns from a spatial data set. Colocation rule mining presents challenges due to the following reasons: First, it is difficult to find colocation instances since spatial objects are embedded in a continuous space and share neighbor relationships. A large fraction of the computation time is devoted to identifying the instances of colocation patterns. Second, it is nontrivial to reuse association rule mining algorithms [5], [6] for colocation pattern mining since, unlike market-basket data, spatial data sets often have no predefined transactions. Thus, a current colocation mining algorithm [7] uses a join-based approach to find colocation instances. Its computational performance suffers, however, due to the large number of joins required as the number of colocation patterns and their instances increases. In this paper, we propose a novel joinless approach for colocation pattern mining which 1) materializes spatial neighbor relationships without any loss of colocation instances and 2) reduces the computational cost of identifying colocation instances using an instance-lookup scheme.

1.1 Related Work

The problem of mining association rules based on spatial relationships (e.g., proximity and adjacency) was first discussed in [6]. The work discovers the subsets of spatial features frequently associated with a specific feature, e.g., cancer. Fig. 1a shows an example data set with three spatial features, A, B, and C. Each object is represented by its feature type and unique instance id, e.g., A.1. Identified neighbor objects are connected by solid lines. Fig. 1b shows the neighbor objects near the objects of a specific reference feature type, A. A set of neighboring objects of each reference object is converted to a transaction. All rules generated from the transactions are related to the specific feature. However, directly applying this approach to colocation mining may not capture our colocation meaning with no specific reference feature.

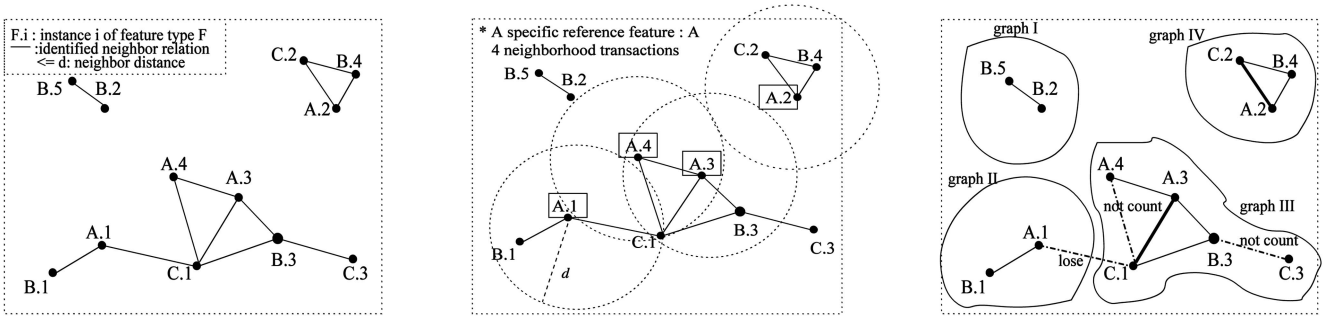


Fig. 1. (a) An example spatial data set. (b) A reference feature-based spatial association rule mining. (c) Subgraph mining.

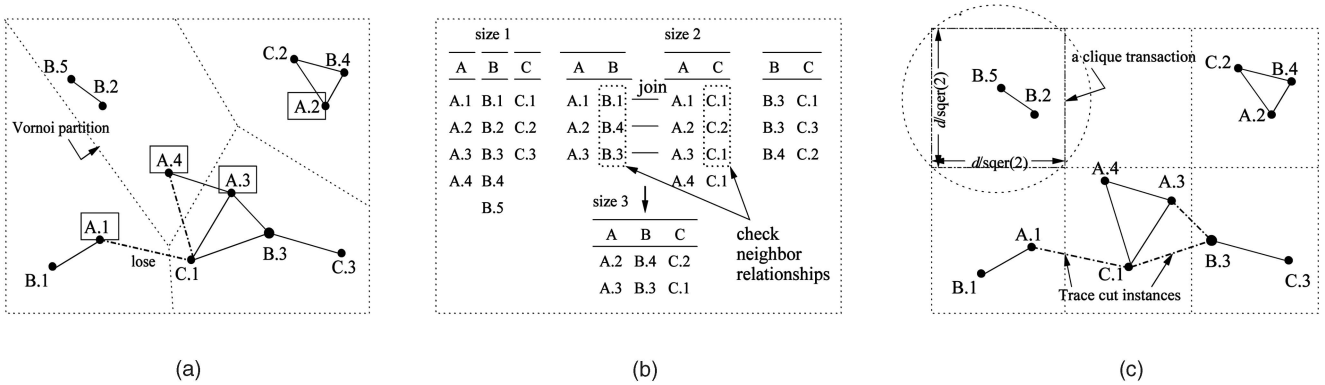


Fig. 2. Different approaches for finding colocation instances. (a) Space partition. (b) Instance join. (c) Clique partition and residual instances.

Previous works on colocation pattern mining have presented different approaches for identifying colocation instances and choosing the interest measures of colocation patterns. Morimoto [8] discovers frequent neighboring class (e.g., colocated features) sets using a *support count* measure. This approach uses a space partitioning and nonoverlap grouping scheme for identifying neighboring objects. However, the explicit space partitioning approach may miss colocation instances across partitions such as {A.4, C.1} in Fig. 2a. Shekhar and Huang [7] proposed statistically meaningful interest measures for colocation patterns and a join-based colocation mining algorithm. Their instance join operation for generating colocation instances is similar to *apriori_gen* [5]. First, after finding all neighbor object pairs (size 2 colocation instances), the approach finds size $k(>2)$ colocation instances by joining the instances of its size $k-1$ subset colocations where the first $k-2$ objects are common and checking the neighbor relationship between the $k-1$th objects. Fig. 2b shows the instances of colocation {A, B, C} being generated. This approach finds correct and complete colocation instance sets. However, the apriori-join operation is computationally expensive with the increase of colocation instances. In the spatial database literature, multiway join techniques using R-trees for multiple spatial features [9], [10] can be used as alternatives to the apriori-join. However, we assume our spatial data set has no spatial index and, thus, it is difficult to apply these techniques to our colocation mining problem directly. In our previous work [11], we proposed a partial join algorithm to reduce the number of expensive join operations in finding colocation instances. This approach transactionizes objects using clique neighbor relationships, e.g., posing simple grids as in Fig. 2c, and uses the apriori-join only for the residual instances not modeled by the explicit transactionization. This method reduces the number of join operations significantly. However, its performance depends on the number of cut instances by explicit partitioning.

Colocation pattern mining may appear to be similar to subgraph mining [12], [13], which finds frequent subgraphs in a large graph database. The prevalent interest measure used is *support*, the ratio of graphs which include a subgraph pattern. For example, in Fig. 1c, the support of subgraph A C is $\frac{2}{4}$. Subgraph mining is a different problem from our colocation mining, which finds all frequent subsets (i.e., clique subgraphs) from a spatial data set (i.e., conceptually, a single graph which represents input spatial objects and their neighbor relationships). A spatial data set can be represented to a set of disjoint graphs and a subgraph mining technique might be applied to it. However, some neighbor relationships can be lost by the disjoint graph partition, e.g., {A.1, C.1} in Fig. 1c, and the support measure may count one instance in a graph, e.g., {A.3, C.1}, but not other instances, e.g., {A.4, C.1}.

1.2 Our Contributions

We took the first step of introducing a joinless algorithm for colocation pattern discovery in [14]. We continue that work here and make the following contributions:

- First, we propose materializing the neighbor relationships of spatial data for efficient colocation pattern mining. We present two novel neighborhood

partition models, star neighborhood partitioning and clique neighborhood partitioning, and compare the models. We also discuss our design decision for colocation instance filtering and prevalent colocation filtering.
- Second, we develop a joinless colocation mining algorithm based on the star neighborhood partition model. The algorithm is efficient since it uses an instance-lookup scheme instead of an expensive spatial or instance join operation for filtering colocation instances. It also has a coarse pruning step which can filter candidate colocations without finding exact colocation instances. We analytically prove our joinless algorithm is correct and complete, i.e., there are no false droppings or false admissions in finding colocation rules.
- Third, we apply the clique partition model to spatial data especially clustered in neighborhood areas and describe the partial join colocation mining algorithm with a new coarse filtering scheme.
- Fourth, we provide algebraic cost models to analyze the performance dominance zones of the joinless method, the partial join method, and a current join-based method [7]. We compare the computational complexity of the different methods.
- Finally, we experimentally evaluate our algorithms using synthetic data sets and real-world data sets. The joinless method outperforms the join-based method in overall parameter settings and shows scalability to dense data.

1.3 Scope and Outline

This paper addresses the colocation pattern mining of spatial point data. We focus on the analytical and experimental comparisons with a current join-based algorithm [7] to examine the effects of our algorithmic design decisions. Discussion of optimized partitioning methods for the clique partition model and discussion of the significance of colocation interest measures are beyond the scope of this paper.

The remainder of the paper is organized as follows: Section 2 gives an overview of the basic concepts of colocation pattern mining and the problem definition. In Section 3, we describe our algorithmic design decision for colocation pattern mining. In Section 4, we present our joinless colocation mining algorithm and the completeness and correctness of the algorithm. In Section 5, we describe the partial join method briefly. The analytical analysis of our algorithms and a current join-based colocation algorithm is given in Section 6. Section 7 presents the experimental evaluation. The conclusion and future work are discussed in Section 8.

2 Colocation Pattern Mining

In this section, we describe the basic concepts of colocation pattern mining and the problem definition.

2.1 Basic Concepts

Given a set of spatial features F, a set of their instance objects S, and a neighbor relationship R over S, a **colocation** C is a subset of spatial features, $C \subseteq F$, whose instances form a clique using a neighbor relationship R. When the neighbor relationship R is a Euclidean distance metric with its threshold value d, two spatial objects are neighbors if they satisfy the neighbor relationship, i.e., $R(\mathrm{A.1}, \mathrm{B.1}) \Leftrightarrow (distance(\mathrm{A.1}, \mathrm{B.1}) \leq d)$. A **colocation instance** I is a set of objects, $I \subseteq S$, which includes the objects of all features in the colocation and forms a clique relationship. In Fig. 1a, {A.2, B.4, C.2} is an instance of colocation {A, B, C} since the feature type of A.2 is A, the feature type of B.4 is B, and the feature type of C.2 is C, and $R(\mathrm{A.2}, \mathrm{B.4})$, $R(\mathrm{A.2}, \mathrm{C.2})$, and $R(\mathrm{B.4}, \mathrm{C.2})$. In this paper, we use the terms "instance" or "clique instance" interchangeably to refer to a colocation instance.

A **colocation rule** is of the form: $C_1 \rightarrow C_2(p, cp)$, where $C_1 \subseteq F$, $C_2 \subseteq F$, and $C_1 \cap C_2 = \emptyset$. The interest of a colocation rule can be measured by its prevalence (p) and conditional probability (cp). We use the colocation interest measures defined by [7]. First, the **participation ratio** $Pr(C, f_i)$ of feature f_i in a colocation $C = \{f_1, \ldots, f_k\}$, $1 \leq i \leq k$, is the fraction of objects of feature f_i in the neighborhood of instances of colocation $C - \{f_i\}$. It is defined as $Pr(C, f_i) = \frac{Number\ of\ distinct\ objects\ of\ f_i\ in\ instances\ of\ C}{Number\ of\ objects\ of\ f_i}$. The **participation index** $Pi(C)$ of a colocation $C = \{f_1, \ldots, f_k\}$ is defined as $Pi(C) = \min_{f_i \in C}\{Pr(C, f_i)\}$. A high participation index of a colocation indicates that the spatial features of the colocation likely show up together. For example, in the data set of Fig. 1a, feature A has four instances, feature B has five instances, and feature C has three instances. Consider the prevalence values of colocation $c = \{\mathrm{A}, \mathrm{B}, \mathrm{C}\}$. The instances of colocation c are {A.2, B.4, C.2} and {A.3, B.3, C.1}. The participation ratio of feature A in the colocation c, $Pr(c, \mathrm{A})$ is $\frac{2}{4}$ since only A.2 and A.3 among four feature A objects are involved in the colocation instances. $Pr(c, \mathrm{B})$ is $\frac{2}{5}$ and $Pr(c, \mathrm{C})$ is $\frac{2}{3}$. Thus, the participation index of colocation c, $Pi(c)$, is $min\{Pr(c, \mathrm{A}), Pr(c, \mathrm{B}), Pr(c, \mathrm{C})\} = \frac{2}{5}$. The **conditional probability** $P(C_1|C_2)$ of a colocation rule $C_1 \rightarrow C_2$ is the fraction of instances of C_2 in the neighborhood of instances of C_1. Formally, it is estimated as

$$P(C_1|C_2) = \frac{Number\ of\ distinct\ instances\ of\ C_1\ in\ instances\ of\ C_1 \cup C_2}{Number\ of\ instances\ of\ C_1}.$$

In the example of Fig. 1a, the conditional probability of colocation rule $A \rightarrow ABC$, $P(A|ABC)$ is $\frac{2}{4}$.

Lemma 1. *The participation ratio and the participation index are monotonically nonincreasing with increases in the size of the colocation.*

For example, the participation index of a size 3 colocation is not greater than the participation index value of any size 2 colocation, e.g., $Pi(\{A, B, C\}) = \frac{2}{5} \leq Pi(\{A, B\}) = \frac{3}{5}$ in Fig. 1a. Please refer to [7] for the proof of Lemma 1.

2.2 Problem Definition

We focus on finding a correct and complete set of colocation rules while reducing the computation cost. The formal problem definition is as follows.

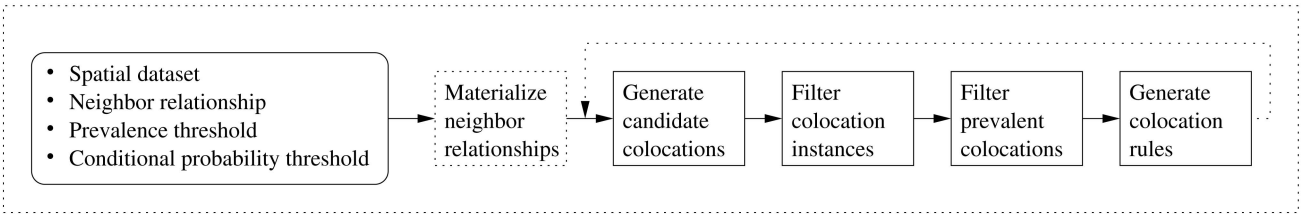


Fig. 3. Basic procedure of collocation pattern mining.

Given:
1) A set of spatial feature types $F = \{f_1, \ldots, f_n\}$.
2) A set of spatial objects $S = S_1 \cup \ldots \cup S_n$, where $S_i (1 \leq i \leq n)$ is a set of instances of feature type f_i. Each object $o \in S_i$ has a vector information of $<$ feature type f_i, instance id j, and location $x, y >$, where $1 \leq j \leq |S_i|$.
3) A spatial neighbor relationship R.
4) A minimum prevalence threshold (min_prev) and a minimum conditional probability threshold (min_cond_prob)

Find:
A set of colocation rules with participation index $\geq min\_prev$ and conditional probability $\geq min\_cond\_prob$.

Objective:
Find a correct and complete set of colocation rules while reducing the computation cost.

Constraints:
R is a distance metric-based neighbor relationship and has a symmetric property.

3 Design Decisions for Colocation Pattern Mining Algorithm

In this section, we present our algorithmic design decisions for the colocation pattern mining.

3.1 Basic Algorithm

Given a spatial data set, a neighbor relationship, and interest measure thresholds, the basic colocation pattern mining procedure involves four steps, as shown in Fig. 3. First, candidate colocation sets are generated and their colocation instances are gathered from the spatial data set. Prevalent colocation sets satisfying the given prevalence threshold are filtered. Finally, colocation rules satisfying the conditional probability threshold are generated. Most of the computational time of colocation pattern mining is devoted to finding colocation instances with clique neighbor relationships. Thus, we propose adding a step for materializing the spatial neighbor relationships to increase the efficiency of colocation mining. In the following subsections, we discuss our design decisions in three areas: neighborhood materialization, colocation instance filtering, and prevalent colocation filtering.

3.2 Neighborhood Materialization

Given a neighbor relationship, a spatial data set can be represented as a neighbor graph in which a node is a spatial object and an edge between two nodes represents the neighbor relationship. Fig. 4a depicts a spatial data set as a neighbor graph. The materialization of neighbor relationships for colocation mining should satisfy two criteria. No neighbor relation should be missed during the materialization process and the computation cost has to be cheap. In addition, it is preferable to minimize duplicate neighbor relations. We propose two neighborhood materialization models, star neighborhood partitioning and clique neighborhood partitioning, and then compare them.

3.2.1 Star Neighborhood Partition Model

The **star neighborhood partition model** (simply, the star partition model) partitions neighbor relationships using a star neighbor relationship. For example, Fig. 4b shows the possible neighborhood areas of star relationships with objects A.1, A.3, and B.4, which are represented by dotted circles whose radii are a user specific neighbor distance. The black solid lines in each circle represent a star neighbor relationship with the center object. For example, A.1 has a star neighbor relationship with B.1 and C.1. The star partition model can be divided into two types, *overlap star partitioning* and *disjoint star partitioning*. Overlap star partitioning results in duplicate neighbor relations, e.g., a neighbor relationship between A.1 and B.1 is duplicated in the star relationship of A.1 and in the star relationship of B.1. In contrast, disjoint star partitioning divides the whole neighbor relationships to a set of disjoint star relations. The disjoint star partition model is preferred since it can lead to no duplication nor any loss of neighbor relationships. The star neighborhood of an object identified by the disjoint star partition model is defined as follows:

Definition 1. *Given a spatial object $o_i \in S$ whose feature type is $f_i \in F$, the **star neighborhood** of o_i is defined as a set of spatial objects $SN = \{o_j \in S | o_j = o_i \vee (f_j > f_i \wedge R(o_j, o_i))\}$, where $f_j \in F$ is the feature type of o_j and R is a neighbor relationship.*

The star neighborhood of an object is a set of the center object and objects in its neighborhood whose feature types are greater than the feature type of the center object in a lexical order. In Fig. 4b, A.1 has two neighboring objects, B.1 and C.1. The star neighborhood of A.1 is {A.1, B.1, C.1}, including the center object A.1. In the case of A.3, three neighboring objects, A.4, B.3, and C.1, are present. However, A.4 is not included in the star neighborhood of A.3 since we focus on colocation patterns among different feature types. Next, B.4 has two neighbor objects, A.2 and C.2. However, A.2 is not included in the star neighborhood of B.4 since the neighbor relationship between A.2 and B.4 is

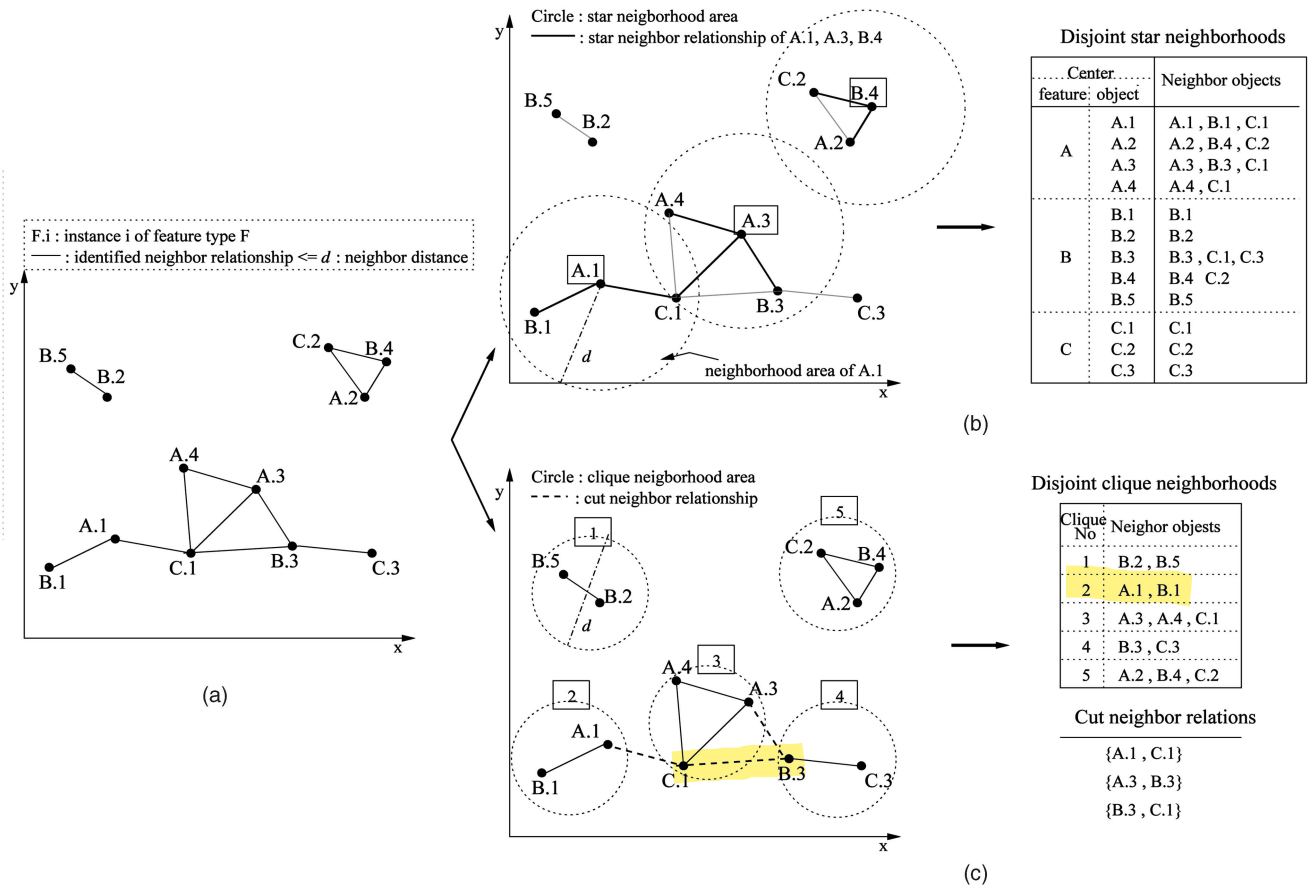


Fig. 4. Neighborhood materialization. (a) Spatial data set (as neighbor graph). (b) Star neighborhood partition. (c) Clique neighborhood partition.

already reflected in the star neighborhood of A.2. All star neighborhoods of the spatial data set are listed in Fig. 4b.

3.2.2 Clique Neighborhood Partition Model

The **clique neighborhood partition model** (simply, the clique partition model) partitions a spatial data set into sets of objects having clique relationships. For example, in Fig. 4c, A.2, B.4, and C.2 have a clique relationship, and become one partition. The clique partition model can also be divided into two types, *overlap clique partitioning* and *disjoint clique partitioning*. Overlap clique partitioning generates a set of maximal cliques. In contrast, disjoint clique partitioning divides a spatial data set into a set of disjoint cliques in which each clique is not always maximal. The disjoint clique partition model is preferred since overlap clique partitioning requires finding all maximal cliques, which is computationally expensive (NP-Complete). The following is the definition of a clique neighborhood by the disjoint clique partition model.

Definition 2. *A **clique neighborhood** is a set of spatial objects $CN \subseteq S$ that forms a clique under a neighbor relationship R. A spatial data set S is a union of disjoint clique neighborhoods, $\{CN_1, \ldots, CN_m\}$, where $CN_i \cap CN_j = \emptyset$, $i \neq j$, and $CN_1 \cup \ldots \cup CN_m = S$.*

Each spatial object belongs to one clique neighborhood exactly. Fig. 4c illustrates the clique partition model. A dashed circle represents a neighborhood area whose radius is $d/2$. The objects within the circle are neighbors of each other. For example, A.1 and B.1 form a clique neighborhood. However, note that this disjoint clique partitioning can lose neighbor relationships across the partitions, e.g., $R(\text{A.1}, \text{C.1})$. Fig. 4c presents cut neighbor relationships as dotted lines. {A.1, C.1}, {A.3, B.3}, and {B.3, C.1} are the cut neighbor relations. Our clique partition model also keeps the cut neighbor relations for completeness. Fig. 4c shows a set of the disjoint clique neighborhoods and cut neighbor relations.

3.2.3 Model Comparison

We compare the star partition model and the clique partition model in terms of their characteristics, the materialized data size, and advantages and disadvantages in colocation pattern mining. First, if we consider the two models from the perspective of a graph, the star partition model (also called the *edge partition model*) partitions edges disjointly using a star neighbor relationship. In contrast, the clique partition model (also called the *node partition model*) partitions nodes disjointly using clique neighbor relationships. The star partition model has n star neighborhoods, where n is the total number of objects in a spatial data set. The number of clique neighborhoods depends on the data distribution. In a declustered spatial data set where each object is one clique neighborhood, the number of clique neighborhoods is n. In general, the number of clique neighborhoods is much smaller than the number of star

neighborhoods. Next, we compare the advantages and disadvantages of the models in colocation pattern mining. Star neighborhoods are easily generated from a neighbor graph. However, a star neighborhood does not model a clique relationship directly and can contain "false positives" of colocation instances. In contrast, the cost to generate clique neighborhoods depends on the partitioning method, e.g., from a simple grid partition to a fine partition depending on data distribution. The ideal case is to maximize the number of objects in a partition and to minimize the number of cut relationships. The benefit of the clique partition model is that it models clique relationships directly. However, this model requires a procedure to trace "false dismissals" split by the disjoint object partition, i.e., colocation instances related to cut relationships.

3.3 Colocation Instance Filtering

The algorithmic design for filtering colocation instances from a spatial data set depends on the neighborhood materialization model. We present two schemes for colocation instance filtering, *instance-lookup* and *instance-join*. We define the following terms to distinguish the instances of colocations from different neighborhood materialization models:

Definition 3. *Let $I = \{o_1, \ldots, o_k\} \subseteq S$ be a set of spatial objects whose feature types $\{f_1, \ldots, f_k\}$ are different. If all objects in I are neighbors to the first object o_1, I is called a* ***star instance*** *of colocation $C = \{f_1, \ldots, f_k\}$.*

For example, in Fig. 4b, a subset of the A.1 star neighborhood, {A.1, B.1} is a star instance of colocation {A, B}. Next, we define terms related to the clique neighborhoods.

Definition 4. *Let $I = \{o_1, \ldots, o_k\} \subseteq S$ be a (clique) instance of colocation C. If all objects in I belong to a same clique neighborhood, I is called an* ***intraneighborhood instance*** *(simply, an intrainstance) of colocation C,*

For example, in Fig. 4c, {A.1, B.1} is an intrainstance of colocation {A, B}.

Definition 5. *A neighbor relationship between two objects, $o_1, o_2 \in S, o_1 \neq o_2$, is called a* ***cut neighbor relationship*** *if o_1 and o_2 are neighbors of each other but belong to different clique neighborhoods.*

Definition 6. *Let $I = \{o_1, \ldots, o_k\} \subseteq S$ be a (clique) instance of a colocation C. If all objects in I have at least one cut neighbor relationship, I is called an* ***interneighborhood instance*** *(simply, an interinstance) of colocation C.*

For example, in Fig. 4c, {A.1, C.1} is an interinstance of colocation {A, C} because A.1 has a cut neighbor relationship with C.1 and C.1 also has cut neighbor relationships with A.1 and B.3. The objects in an interinstance are located across one more clique neighborhoods.

3.3.1 Instance-Lookup

We propose an instance-lookup scheme to filter colocation instances from star instances. A star instance $\{o_1, \ldots, o_k\}$ of colocation $\{f_1, \ldots, f_k\}$ can be gathered from star neighborhoods whose center object type is f_1 by Definition 3. Note that the star instance is not a colocation instance since it is not guaranteed that it has a clique relationship. However, if all objects in it except the first object (the center object of the star relationship) form a clique, the star instance is a colocation instance. The cliqueness of the subset can be checked by searching other star instances, e.g., the second object o_2's star instance, $\{o_2, \ldots, o_k\}$, the third object o_3's star instance, $\{o_3, \ldots, o_k\}$, etc., or by looking up all the clique instances of colocation $\{f_2, \ldots, f_k\}$ except f_1.

3.3.2 Instance-Join

Intraneighborhood instances (i.e., instances within neighborhoods) can be gathered by scanning the clique neighborhood set. An intrainstance is a colocation instance since a subset of a clique neighborhood is also a clique. In contrast, interneighborhood instances (i.e., instances over different neighborhoods) are not found from the materialized set directly. We adopt an instance join method [7] for tracking the interinstances. The interinstances of a size k colocation can be generated by joining the interinstances of its size $k-1$ colocations. For example, in Fig. 4c, an interinstance {A.3, B.3, C.1} can be generated by joining size 2 interinstances, {A.3, B.3} and {A.3, C.1}, and checking the neighbor relationship between B.3 and C.1.

3.4 Colocation Filtering

We use three filtering steps to find prevalent colocations: *feature-level filtering*, *coarse filtering*, and *refinement filtering*. Feature-level filtering is possible due to the antimonotonic property of the participation index by Lemma 1. A candidate colocation can be pruned without examining its instances if any subset of it is not prevalent. Coarse filtering is done after finding all star instances or after finding all intrainstances of colocations, which depends on the materialization model. This step can reduce the expensive filtering operation for colocation instances. If the prevalence value of star instances of a candidate colocation does not satisfy a given threshold, the candidate is pruned without the cliqueness check of its star instances. In the clique partition model, the intrainstances of a candidate colocation do not give enough information for the coarse pruning. Thus, we use cardinal objects from the interinstances of its subsets as well as its intrainstances. If the prevalence value from them does not satisfy the threshold, the colocation is pruned without finding its interinstances. In the last step, the refinement filtering decides prevalent colocation sets after filtering all colocation instances.

3.5 Design Decision

The choice of neighborhood materialization model can depend on the distribution of the input spatial data. For general spatial data sets, we choose the star neighborhood partition model and use the instance lookup scheme for filtering colocation instances. We present the joinless colocation mining algorithm in the next section. In contrast, the clique partition model is appropriate for spatial data mostly clustered in neighborhood areas having fewer neighbor relationships with their nearby neighborhoods. This is because clustered data can be modeled to clique neighborhoods and fewer neighbor interactions with nearby neighborhoods can reduce the number of cut relationships. The partial join approach is briefly discussed in Section 5.

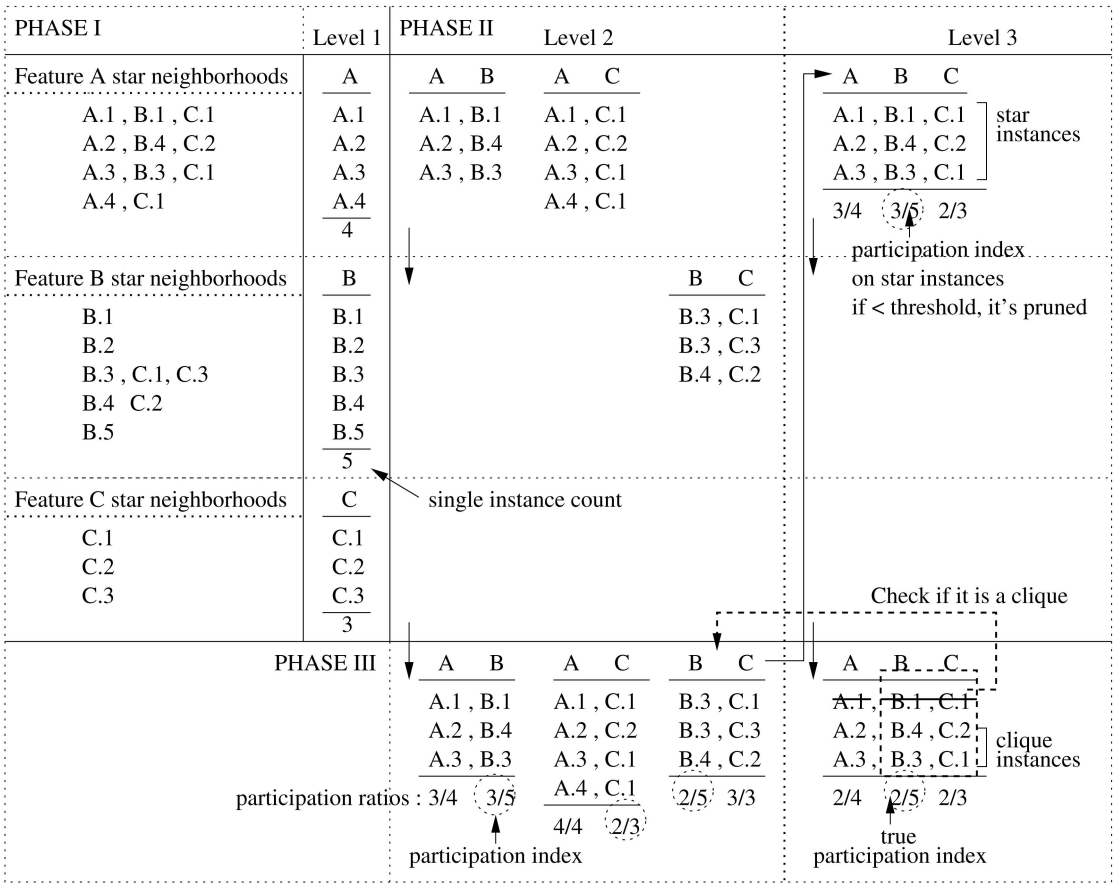


Fig. 5. An illustration of the joinless colocation mining algorithm.

4 Joinless Approach

In this section, we present a *joinless colocation mining algorithm* based on the star neighborhood partition model. We show the algorithm is correct and complete in mining colocation patterns.

4.1 Joinless Colocation Mining Algorithm

The joinless colocation mining algorithm has three phases. The first phase converts an input spatial data set into a set of disjoint star neighborhoods. The second phase gathers the star instances of candidate colocations from the materialized set and coarsely filters the candidates using the prevalence values of their star instances. The third phase filters their colocation instances from the star instances and finds prevalent colocations and generates colocation rules. The second and third phases are repeated with the increase of colocation size. Fig. 5 illustrates a trace example of the joinless algorithm. Algorithm 1 shows the pseudocode. The detailed explanation is given as follows:

Algorithm 1 Joinless collocation mining algorithm
Inputs
$F = \{f_1, \ldots, f_n\}$: a set of spatial feature types
S: a set of spatial objects
R: a spatial neighbor relationship
min_prev: prevalence threshold,
min_cond_prob: conditional probability threshold
Output
A set of all colocation rules with participation index $\geq$ min_prev and conditional probability $\geq$ min_cond_prob
Variables
$SN = \{SN_{f_1}, \ldots, SN_{f_n}\}$
:a set of feature f_i star neighborhoods
k: colocation size
C_k: a set of size k candidate colocations
SI_k: a set of star instances of size k candidates
CI_k: a set of clique instances of size$k$ candidates
P_k: a set of size k prevalent colocations
R_k: a set of size k colocation rules
Method
1) $SN = \texttt{gen\_star\_neighborhoods}(F, S, R)$;
2) $P_1 = F$; $k = 2$;
3) while (not empty P_{k-1}) do
4) $C_k = \texttt{gen\_candidate\_colocations}(P_{k-1})$;
5) for i in 1 to n do
6) for $t \in SN_{f_i}$ where $f_i = cf_1$, cf_1 is the first feature of $C_k(cf_1, \ldots, cf_k)$.
7) $SI_k = \texttt{filter\_star\_instances}(C_k, t)$;
8) end do
9) if $k = 2$ then $CI_k = SI_k$
10) else do $C_k = \texttt{select\_coarse\_prevalent\_colocations}(C_k, SI_k, min\_prev)$
11) $CI_k = \texttt{filter\_clique\_instances}(C_k, SI_k)$;
12) end do
13) $P_k = \texttt{select\_prevalent\_colocations}(C_k, CI_k, min\_prev)$;

14) $R_k = \texttt{gen\_colocation\_rules}(P_k, min\_cond\_prob)$;
15) $k = k + 1$;
16) `end do`
17) `return` $\bigcup(R_2, \ldots, R_k)$;

4.1.1 Convert a Spatial Data Set to a Set of Disjoint Star Neighborhoods (Step 1)

Given an input spatial data set and a neighbor relationship, first find all neighboring object pairs using a geometric method such as plane sweep [15]. The star neighborhoods can be generated from the neighbor pairs by grouping the neighbor objects per each object. Fig. 5 shows the same star neighborhood set in Fig. 4b.

4.1.2 Generate Candidate Colocations (Step 4)

First, we initialize all features to size 1 prevalent colocations by the definition of participation index. The number of instances per each feature can be known during the neighborhood materialization procedure. Size k $(k > 1)$ candidate colocations are generated from prevalent size $k-1$ colocations (Step 4). Here, we have the feature level filtering of the colocations. If any subset of a candidate colocation is not prevalent, the candidate colocation is pruned.

4.1.3 Filter the Star Instances of Candidate Colocations from the Star Neighborhood Set (Step 5)

The star instances of a candidate colocation are gathered from the star neighborhoods whose center object's feature typ is the same as the first feature of the colocation. For example, the instances of colocation {B, C} are gathered from the feature B star neighborhoods and the instances of colocation {A, B, C} are gathered from the feature A star neighborhoods. Notice that the number of candidate colocations examined in each star neighborhood is much smaller than the number of actual candidates.

4.1.4 Select Coarsely Prevalent Colocations Using Their Star Instances (Step 10)

The size 2 star instances are clique instances since our neighbor relationship is symmetric. Thus, we go to Step 13 to find prevalent colocations. For size 3 or more, we need to check if the star instances are clique instances. Before this procedure, we have a coarse filtering of the colocations. We filter a candidate colocation using the participation index of its star instances. For example, in Fig. 5, the participation index of candidate colocation {A, B, C} from its star instances is $\frac{3}{5}$. If it is less than a user specified minimum prevalent threshold, the candidate is pruned without examining exact colocation instances.

4.1.5 Filter Colocation Instances (Step 11)

From the star instances of a candidate colocation, we filter its colocation instances using the instance look-up scheme. For example, in Fig. 5, to check the cliqueness of a star instance {A.1, B.1, C.1} of colocation {A, B, C}, we examine if a subinstance {B.1, C.1} except A.1 is in the set of clique instances of colocation {B, C}. This instance look up operation can be performed efficiently by an instance key which is composed of the ids of objects in the instance. As shown in Fig. 5, {A.1, B.1, C.1} is not a colocation instance, but {A.2, B.4, C.2} and {A.3, B.3, C.1} are colocation instances.

4.1.6 Select Prevalent Colocation Sets (Step 13)

The refinement filtering of colocations is done by the true participation index values calculated from their colocation instances. Prevalent colocations satisfying a given threshold are selected.

4.1.7 Generate Colocation Rules (Step 14)

Finally, we generate all colocation rules satisfying a given minimum conditional probability from a set of prevalent colocations. Steps 3-16 are repeated as the colocation size increases.

4.2 Completeness and Correctness

Completeness means the joinless algorithm finds all colocation rules whose participation index and conditional probability satisfy a user specified minimum prevalence threshold and conditional probability threshold. Correctness means that all colocation rules generated by the joinless algorithm have participation index and conditional probability values above the minimum thresholds. First, we provide related lemmas.

Lemma 2. *The star partition model does not miss any neighbor relationship of an input spatial data set.*

Proof. The disjoint star partition model includes all neighbor relations of each object and excludes only duplicate neighbor relations which are already included in a star neighborhood by Definition 1. □

Lemma 3. *Let $C = \{f_1, \ldots, f_k\}$ be a size k colocation and SI be the star instances of C. The participation index of C from SI is not less than the true participation index of C.*

Proof. The participation ratio of f_1 from SI is the maximum possible probability that the objects of feature f_1 have clique relationships with the objects of the other features $f_2, \ldots, f_k$ in C since only objects of feature f_1 in the star instances can be included in the clique colocation instances of C. The participation ratio of $f_j (2 \le j \le k)$ from SI is also the maximum possible probability that the objects of feature f_j have clique relationships with the objects of features f_1 since our neighbor relationship is symmetric. Thus, the participation index of C calculated from the star instances is not less than the true participation index of C, i.e.,

$$\min_{f_i \in C}\{possible\ max\ Pr(C, f_i)\} \ge \min_{f_i \in C}\{Pr(C, f_i)\}$$

□

Lemma 4. *Let an instance $I = \{o_1, \ldots, o_k\}$ be a star instance of colocation $C = \{f_1, \ldots, f_k\}$. If the subinstance $\{o_2, \ldots, o_k\}$ except o_1 is a clique, I is a colocation instance of C.*

Proof. In a star instance $I = \{o_1, \ldots, o_k\}$, the first object o_1 has neighbor relationships to the other objects, $o_2, \ldots, o_k$, by Definition 3. If a subset $\{o_2, \ldots, o_k\}$ is a clique, object $o_j (2 \le j \le k)$ has neighbor relationships with all objects in the subset as well as o_1. Thus, I is a clique colocation instance. □

Theorem 1. *The joinless colocation mining algorithm is complete.*

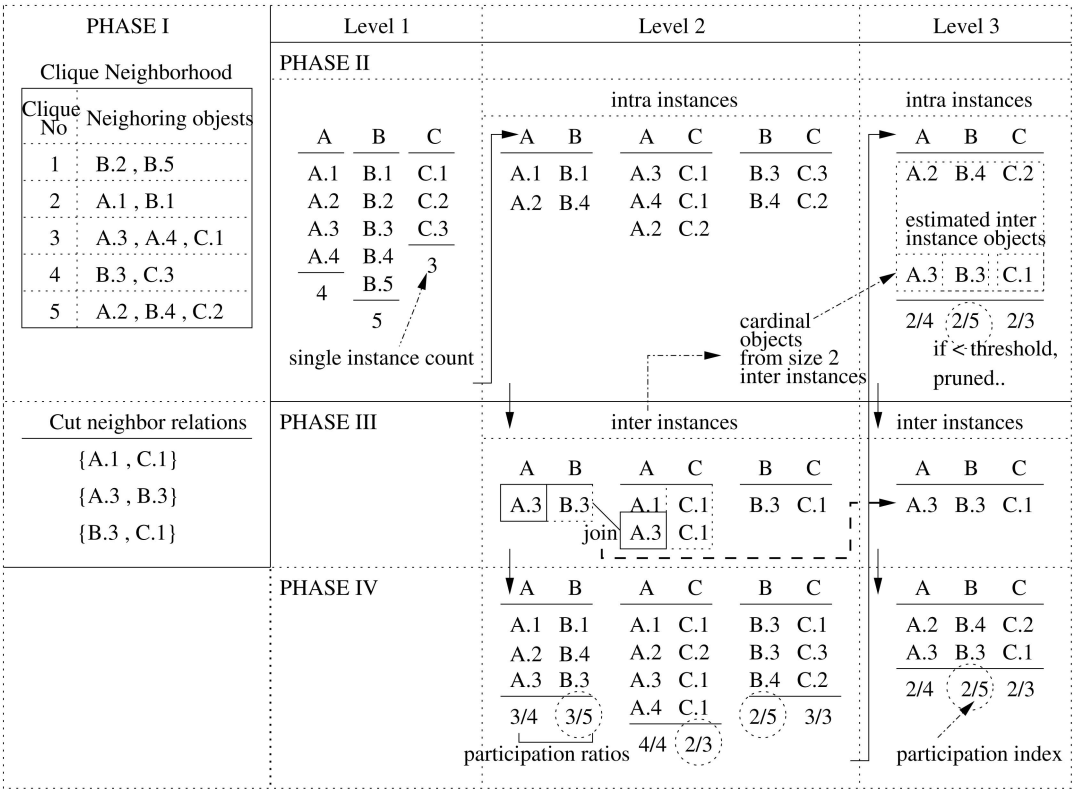


Fig. 6. An illustration of the partial join colocation mining algorithm.

Proof. The completeness of the joinless algorithm can be shown in two ways. The first is that the method to materialize the neighbor relationships of an input spatial data set (Step 1), the method to gather star instances (Step 7), and the method to filter clique instances (Step 11) are correct. The star partition model does not miss any neighbor relationship of the input data by Lemma 2. The star instances gathered from the star neighborhoods where the feature type of the center object is the same as the first feature of the colocation have correct star neighbor relationships by Definitions 1 and 3. No potential colocation instance is missed since the star instances are a super set of the clique instances. The method to filter colocation instances from the star instances does not drop a true instance by Lemma 4. Next, we show that the filtering steps of colocations do not miss true colocations. The feature level filtering by prevalent subsets (Step 4) is complete by Lemma 1. The coarse filtering (Step 10) does not eliminate any true prevalent colocations by Lemma 3. The refinement filtering (Step 13) prunes only colocations whose true participation index is less than the threshold. Step 14 ensures that no colocation rules satisfying a given conditional probability are missed. □

Theorem 2. *The joinless colocation mining algorithm is correct.*

Proof. The correctness of the joinless algorithm can be guaranteed by Steps 13 and 14. Step 13 selects only colocations whose participation indexes satisfy a user specific prevalence threshold. The generated colocation rules by Step 14 also satisfy a user-specific conditional probability. □

5 Partial Join Approach

In this section, we briefly describe a partial join approach for spatial data sets that are mostly clustered in neighborhood areas with fewer neighbor relationships with their nearby neighborhoods. The *partial join colocation mining algorithm* was proposed in our previous work [11]. Here, we have added a coarse filtering scheme to further reduce the number of join operations. The partial join approach uses the clique partition model for materializing clustered data to clique neighborhoods directly and keeps track of colocation instances split by the disjoint object partitioning. Fig. 6 shows a trace example of the partial join colocation mining algorithm.

Given a spatial data set and a neighbor relationship, first we need to generate a set of disjoint cliques. To do this, we use grid partitioning. Rectangular grids of size $d/\sqrt{2} \times d/\sqrt{2}$, where d is a neighbor distance, are posed on a spatial framework. The objects in each grid are generated to a clique neighborhood. Cut neighbor relations can be detected by examining neighbor pairs whose partition areas are different. Size 2 interinstances (i.e., instances across neighborhoods) are the cut relations and the pairs of objects having at least one cut neighbor relationship in each partition. The intrainstances (i.e., instances within neighborhoods) of a size k ($k > 2$) candidate colocation are enumerated from the clique neighborhood set. For example, in Fig. 6, {A.2, B.4, C.2} is an intrainstance of colocation {A, B, C}. Before finding the interinstances of a candidate colocation, we conduct a coarse filtering of the colocation using its intrainstances and the cardinal objects from the interinstances of its size

$k-1$ colocations. For example, in Fig. 6, the cardinal objects which can be involved in the interinstances of colocation {A, B, C} are A.3 of feature A, B.3 of feature B, and C.1 of feature C. If the participation index calculated from them does satisfy a given threshold value, we go further to find the exact interinstances. We use an instance join method [7] for generating the interinstances. For example, the interinstances of {A, B, C} are produced by joining an interinstance of {A, B}, {A.3, B.3}, and an interinstance of {A, C}, {A.3, C.1}, and by examining the neighbor relationship between B.3 and C.1. The true participation index of a colocation is calculated from its intrainstances and interinstances. All colocation rules whose prevalence value and conditional probability satisfy given thresholds are generated.

6 Analytical Comparison

In this section, we analytically compare our joinless and partial join colocation algorithms with a pure join-based colocation algorithm [7]. First, we examine the computational complexities of the different methods and then compare them.

6.1 Computational Complexities

Let T_{jl}, T_{pj}, and T_{jb} be the total computation cost of the joinless method, the partial join method, and the join-based method, respectively. The following equations show the total cost functions:

$$\begin{aligned}
T_{jl} &= T_{star_neighborhoods}(S) + T_{jl}(2) + \sum_{k>2} T_{jl}(k),\\
T_{pj} &= T_{clique_neighborhoods\&cut_relations}(S) + T_{pj}(2)\\
&\quad + \sum_{k>2} T_{pj}(k),\\
T_{jb} &= T_{neighbor_pairs}(S) + T_{jb}(2) + \sum_{k>2} T_{jb}(k).
\end{aligned}$$

S denotes an input spatial data set. $T_{jl}(2)$, $T_{pj}(2)$, and $T_{jb}(2)$ represent the cost for finding size 2 colocation patterns in each method. The following equations are the costs of finding size $k(k>2)$ colocation patterns.

$$\begin{aligned}
T_{jl}(k) &= T_{gen_candi}(P_{k-1}) + T_{filter_star_inst}(C_k)\\
&\quad + T_{filter_coarse_coloc}(C_k) + T_{filter_clique_inst}(C'_k)\\
&\quad + T_{filter_prev_coloc}(C'_k) + T_{gen_rules}(P_k)\\
&\approx T_{filter_star_inst}(C_k) + T_{filter_clique_inst}(C'_k).
\end{aligned}$$

$$\begin{aligned}
T_{pj}(k) &= T_{gen_candi}(P_{k-1}) + T_{filter_intra_inst}(C_k)\\
&\quad + T_{filter_coarse_coloc}(C_k) + T_{filter_inter_inst}(C''_k)\\
&\quad + T_{filter_prev_coloc}(C''_k) + T_{gen_rules}(P_k)\\
&\approx T_{filter_intra_inst}(C_k) + T_{filter_inter_inst}(C''_k).
\end{aligned}$$

$$\begin{aligned}
T_{jb}(k) &= T_{gen_candi}(P_{k-1}) + T_{filter_inst}(C_k)\\
&\quad + T_{filter_prev_coloc}(C_k) + T_{gen_rules}(P_k)\\
&\approx T_{filter_inst}(C_k).
\end{aligned}$$

P_{k-1} is a size $k-1$ prevalent colocation set, C_k is a size k candidate colocation set and each C'_k and C''_k is a size k candidate colocation set filtered by the coarse filtering in the joinless algorithm and the partial join algorithm. In these equations, T_{gen_rules} is the same cost in all three methods. T_{gen_candi}, $T_{filter\_coarse\_coloc}$, and $T_{filter\_prev\_coloc}$ can be ignored when compared with other computation factors as shown in the experiment results of the next section.

6.2 Comparison of Computational Complexities

We compare the computational complexities of the different methods in two parts. One is the computation cost of materializing neighborhoods and generating size 2 colocation patterns and the other is the computational cost of generating size $k(k>2)$ colocation patterns.

6.2.1 Neighborhood Materialization and Size 2 Colocations

The computational costs of materializing neighborhoods and generating size 2 colocation patterns in the different methods are compared by the following equation:

$$\begin{aligned}
&T_{clique_neighborhoods\&cut_relations}(S) + T_{pj}(2)\\
&\quad > T_{star_neighborhoods}(S) + T_{jl}(2)\\
&\quad > T_{neighbor_pairs}(S) + T_{jb}(2).
\end{aligned}$$

First, each method needs to find all of the neighboring pairs. The cost is the same in the three methods. The join-based method generates size 2 colocations by gathering the neighbor pairs per candidate colocation and calculating their prevalence measures. In both the joinless method and the partial join method, the size 2 colocations can be found also from the neighbor pairs or by scanning their materialized neighborhood set. If we use the former, the cost is the same in all three methods. However, the joinless method has an additional cost to materialize the star neighborhoods from the neighbor pairs. In the partial join method, we generate the disjoint clique neighborhoods using a simple grid partitioning. The cut relations are also easily found during the neighbor pair search, but an additional cost is required to find all size 2 interinstances with cut relationships. The overall cost is expected to be a little bigger than the cost to generate the star neighborhood set.

6.2.2 Joinless versus Join-Based with Size $k(k>2)$ Colocations

We compare the joinless algorithm with the join-based algorithm in generating size $k(k>2)$ colocation patterns. Equation (1) shows the computation ratio:

$$\begin{aligned}
\frac{T_{jl}(k)}{T_{jb}(k)} &\approx \frac{T_{filter_star_inst}(C_k) + T_{filter_clique_inst}(C'_k)}{T_{filter_inst}(C_k)}\\
&\approx \frac{|C_k| \times t_{scan} + p_{jl} \times |C_k| \times t_{lookup}}{|C_k| \times t_{join}}\\
&\approx \frac{t_{scan} + p_{jl} \times t_{lookup}}{t_{join}}.
\end{aligned} \tag{1}$$

t_{scan} is the average cost to collect the star instances of a candidate colocation by scanning the materialized neighborhood set. p_{jl} is the coarse filtering ratio. t_{lookup} is the average cost to check the cliqueness of its star instances per colocation. t_{join} is the average cost to generate colocation instances using the instance join. If the input data set is dense, $t_{scan} \ll t_{join}$ and the ratio is $\frac{p_{jl} \times t_{lookup}}{t_{join}}$. We assume that the

TABLE 1
Experimental Parameters and Their Values in Each Experiment

Symbols	Meaning	Experiment no. 1-1	1-2	1-3	1-4	2-1	2-2	2-3	2-4	3-1	3-2
S	Average size of †colocations	5									
P	Number of †colocations	20									
I	Average number of †colocation instances	150									
F	Number of features	20				*		20			
N	Number of points	15K			*		15K				
D	Spatial frame size($D \times D$)	*		5000		10000, 1000		5000		10000	
d	Neighborhood distance threshold	10						*		10	
θ	Prevalence threshold	0.3		0.2		0.3			*	0.3	
clumpy	Number of colocation instances generated in a same neighborhood area	1								*	15
across	Ratio of instances across different neighborhood areas over total instances	0								10	*
overlap	Ratio of points overlapped in different colocation instances over total points	0		*		0					

† : initial core colocation, * : variable values

number of instances involved in the join operation and the number of instances involved in the lookup operation per colocation is similar. Even if the coarse pruning is not effective ($p_{jl} = 1$), the joinless algorithm is expected to take less computation time than the join-based algorithm since $t_{lookup} < t_{join}$. We examine the efficiency of t_{lookup} over t_{join} with different density data sets later in the experimental section. With the increase of neighbor density, the number of instances involved in the join operation or the lookup operation increases and the computation difference between t_{lookup} and t_{join} is expected to be bigger. The overall data density is controlled by spatial frame size, and the coarse filtering ratio p_{jl} is controlled by an *overlap* parameter, which is the ratio of the number of points included in different colocation instances over the total number of points.

6.2.3 Partial Join versus Join-Based with size k ($k > 2$) Colocations

$$\begin{aligned}\frac{T_{pj}(k)}{T_{jb}(k)} &\approx \frac{T_{filter\_intra\_inst}(C_k) + T_{filter\_inter\_inst}(C_k'')}{T_{filter\_inst}(C_k)} \\ &\approx \frac{|C_k| \times t_{scan}(intraI) + p_{pj} \times |C_k| \times t_{join}(interI)}{|C_k| \times t_{join}(I)} \\ &\approx \frac{t_{scan}(intraI) + p_{pj} \times t_{join}(interI)}{t_{join}(I)}.\end{aligned} \tag{2}$$

In (2), $intraI$ is the intrainstances from the clique neighborhoods, $interI$ is the interinstances joined in the partial join method and I is the instances joined in the join-based method. In general, $t_{scan}(intraI) \ll t_{join}(I)$ since the cost of gathering instances by scanning a neighborhood materialization set is much cheaper than the join operation for generating whole instances. The number of interinstances in the partial join method is a major factor in this ratio. If $|interI| \ll |I|$, the partial join method is expected to have less computation cost. If $|interI| \simeq |I|$ and the coarse pruning is not effective ($p_{pj} = 1$), $t_{join}(interI) \simeq t_{join}(I)$ and $t_{scan}(intraI)$ cost is added to the partial join method. The join-based method is expected to perform a little better. To control the number of interinstances in the experiment, we use an $across$ ratio, which is the ratio of instances across different neighborhood areas over total instances.

6.2.4 Joinless versus Partial Join with Size $k(k > 2)$ Colocations

$$\begin{aligned}\frac{T_{jl}(k)}{T_{pj}(k)} &\approx \frac{T_{filter\_star\_inst}(C_k) + T_{filter\_clique\_inst}(C_k')}{T_{filter\_intra\_inst}(C_k) + T_{filter\_inter\_inst}(C_k'')} \\ &\approx \frac{p_{jl} \times |C_k| \times t_{lookup}(I)}{p_{pj} \times |C_k| \times t_{join}(interI)} \\ &\approx \frac{p_{jl} \times t_{lookup}(I)}{p_{pj} \times t_{join}(interI)}.\end{aligned} \tag{3}$$

In dense data, $T_{gather\_star\_inst}(C_k)$ and $T_{gather\_intra\_inst}(C_k)$ are expected to be relatively smaller than other factors. If we assume the coarse filter ratios are the same, the comparison of the two methods is the same as the comparisons of T_{lookup} and T_{join} and of $|I|$ and $|interI|$. In dense data sets in neighborhoods with fewer interinstances, the partial join method is expected to show better performance. In the experiment presented later, we compare the behavior of the two algorithms in data density in neighborhood area and the ratio of cut instances. Each factor is controlled by *clumpy* degree and *across* ratio variables.

7 Experimental Evaluation

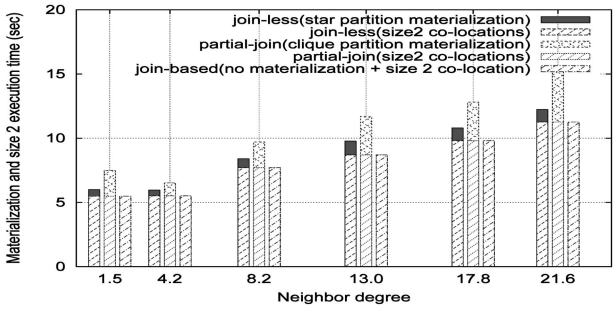
We evaluate the joinless and the partial join colocation algorithms with the join-based algorithm [7] using synthetic and real data sets. We conducted the following experiments:

- We examined the effects of our algorithm design decisions. Specifically, we compared the computation costs of generating up to size 2 colocation patterns with and without a materialization step. We examined the efficiency of the instance-lookup scheme and the effect of coarse filtering. In addition, we compared the costs of the computation complexity factors of the three algorithms to support our analytical comparisons in Section 6.
- We examined the scalability of the joinless algorithm in terms of the number of points, the number of features, the distance neighbor threshold, and the prevalence threshold.

method	Join−less		partial join		join−based	
factor(%) \ data type	sparse	dense	sparse	dense	sparse	dense
T gen_candi	0.0001	0.08	0.0001	0.08	0.0001	0.04
T filter_coarse_coloc	1	2	0.0001	0.02	–	–
T filter_prev_coloc	1	2	0.0001	0.02	0.0001	0.03
T filter_star_inst	60	33	–	–	–	–
T filter_clique_inst	38	62.2	–	–	–	–
T filter_intra_inst	–	–	35	1.67	–	–
T filter_inter_inst	–	–	65	98.18	–	–
T fiter_inst	–	–	–	–	99.99	99.93
total execution time(Sec)	7.19	35.95	8.42	73.09	11.15	111.4

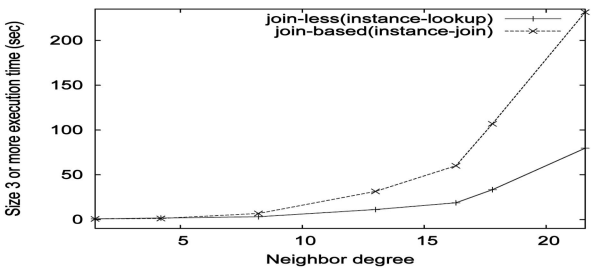
* Execution time is sec. Other data values are %.

(a)

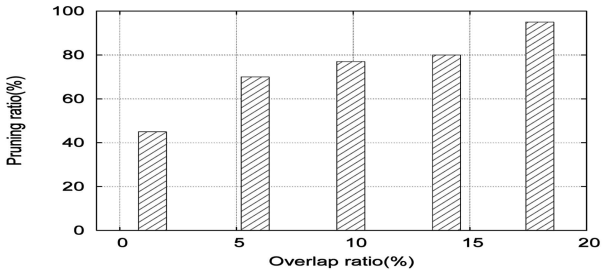


(b)

Fig. 7. (a) Computation of computational complexity factors. (b) Computation comparison of neighborhood materialization and size 2 colocation generation.



(a)



(b)

Fig. 8. (a) Comparison of instance filtering schemes. (b) Effect of coarse pruning in the joinless method.

- We examined the behavior of the partial join algorithm with different neighborhood densities and cut relation ratios.
- Finally, we evaluated the algorithms with two real data sets, a climate data set and an Ecology data set.

All algorithms were implemented in C/C++ and are memory-based algorithms. All the experiments were performed on a Sun SunBlade 1500 with 1.0 GB memory and 177MHz CPU.

7.1 Synthetic Data Generation

Synthetic data sets were generated using a spatial data generator similar to [7]. Table 1 describes the parameters used for the data generation. First, overall spatial frame size was determined with frame side length D. A $D \times D$ size frame was used. The whole frame was divided into regular grids whose side length was a neighborhood distance d. In order to generate colocation instances located across the grids, we divided each grid into two areas, a nonoverlapping area and an overlapping area. An object in the nonoverlapping area has neighbor relationships only with other objects in the same area. By contrast, objects in the overlapping area can have neighbor relationships with objects in the nearby grids.

The following is the procedure we used to generate general synthetic data sets in which colocation instances were randomly distributed. First, we generated P initial patterns whose average size was S. The feature types of each pattern were randomly chosen from F features. An average of I instances per pattern were generated. The total number of points was N. For locating a colocation instance, first we chose a grid cell randomly. All points of the instance were randomly located within the chosen grid cell.

To generate our specialized data sets, first, overall data density was controlled by spatial frame size $D \times D$. Under a fixed total number of points, a smaller frame generates more dense data. Second, the data density in neighborhood areas was controlled by a $clumpy$ degree. When a new grid cell was chosen randomly for locating a colocation instance, a $clumpy$ number of instances was generated in the same grid cell. The default $clumpy$ value was 1. Third, the $across$ ratio controlled the number of instances across different grid cells. An across instance was generated in the overlapping area of a chosen grid and its nearby grid. Finally, the number of points overlapped in different colocation instances was controlled by the $overlap$ ratio. $N \times overlap$ points were randomly selected. Each $overlap$ point was involved in four colocation instances. The parameter values for the synthetic data set used in each experiment are described in Table 1.

7.2 Experiment Results

We present a summary of our experiment results.

7.2.1 Effects of Algorithm Design Decision

Comparison of computational complexity factors. We examined the costs of the computational complexity factors discussed in Section 6. We used two data sets of different density, i.e., a sparse data set with neighbor degree 5 and a dense data set with neighbor degree 15. In Fig. 7a, data values except total execution time represent percent values. Overall, we can notice that T_{gen_candi}, $T_{filter\_coarse\_coloc}$, and $T_{filter\_prev\_coloc}$ take a much smaller portion than the other costs. In

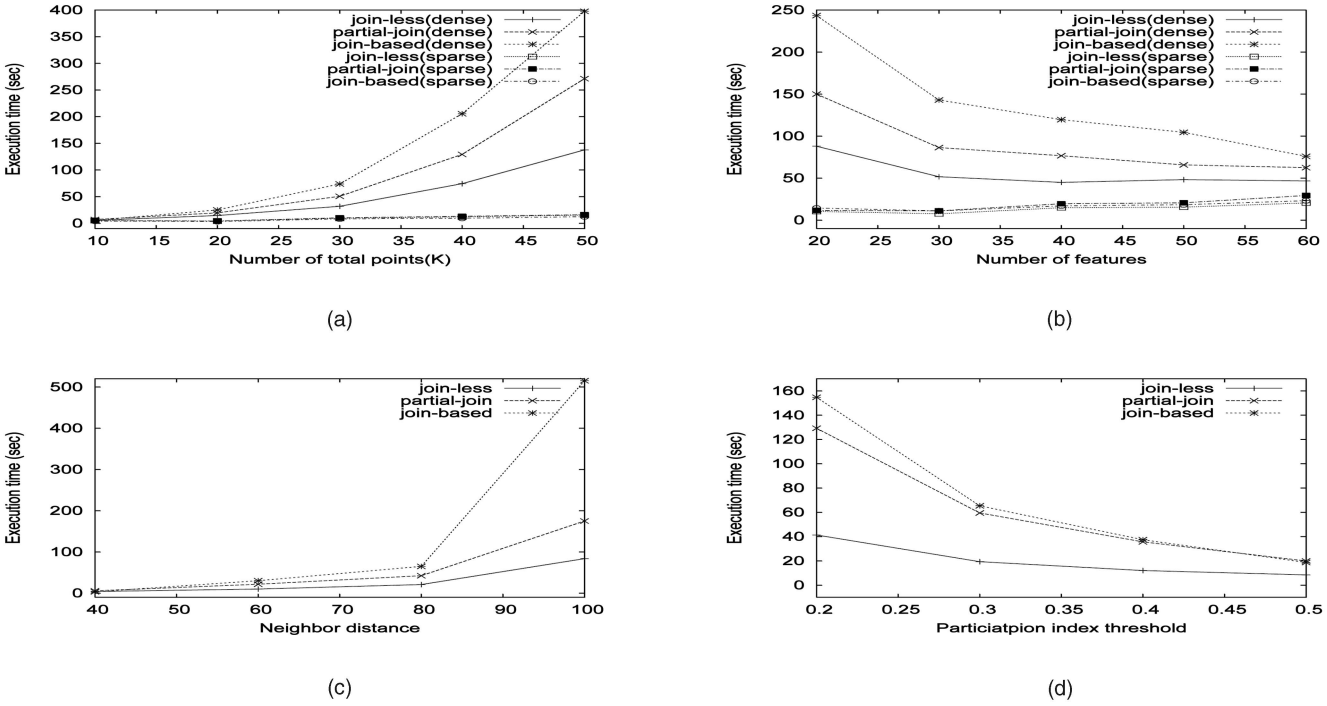


Fig. 9. Scalibility of the joinless algorithm (a) by number of points, (b) by number of features, (c) by neighbor distance, and (d) by prevalence threshold.

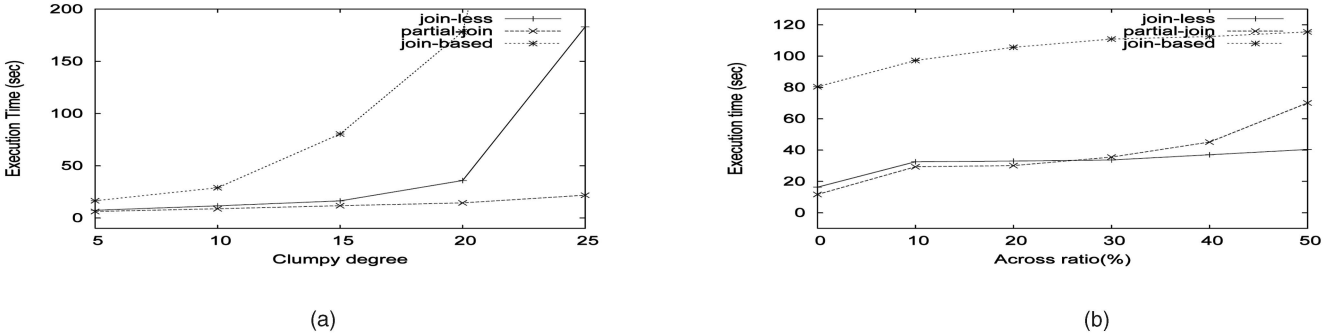


Fig. 10. The effectiveness of the partial join algorithm (a) by data density on a neighborhood area and (b) by cut neighbor relations.

addition, $T_{filter\_star\_inst}$ and $T_{filter\_intra\_inst}$ of gathering instances from each materialization set are relatively smaller than $T_{filter\_inter\_inst}$ and T_{filter_inst} using instance join operations. The difference is greater in the dense data set.

Comparison of neighborhood materialization methods. We compared the computation cost of generating up to size 2 colocations, including the costs of neighborhood materialization. Size 2 colocations were generated from neighboring object pairs in all methods. Fig. 7b shows the execution times of three approaches: with star partition materialization, with clique partition materialization, and with no materialization. As can be seen, the joinless approach without materialization shows less execution time. In the materialization methods, the star partitioning of the joinless approach took less computation time than the clique partitioning of the partial join approach.

Comparison of instance filtering methods. We examined the efficiency of our instance look-up scheme over the instance join method in the join-based algorithm with different dense data sets. Data density determines the neighbor degree of an object, which in turn affects the number of colocation instances. Neighbor degree is the average number of neighbor points per each point. Different spatial frame sizes were used to control the overall data density of the same number of points. Fig. 8a compares the execution time of finding size 3 or more colocation instances of the different schemes. We turned off the coarse filtering option to see only the effects of instance filtering schemes. In sparse data sets (e.g., neighbor degree $<$ 10), the two methods show similar execution time. However, the execution time increases as the neighbor degree increases and, here, the instance-lookup method shows better performance than the instance-join method. Moreover, the difference in execution times of the two methods is much greater with the increase of neighbor degree.

Effect of coarse pruning. We examined the effect of the coarse filtering scheme in the joinless algorithm. Fig. 8b shows the percent of candidate colocations pruned in the coarse filtering step over the total candidate colocations after the feature-level pruning. We considered only size 3 or more candidates since the coarse pruning is effective in filtering them. With the increase of overlap ratio, the chance

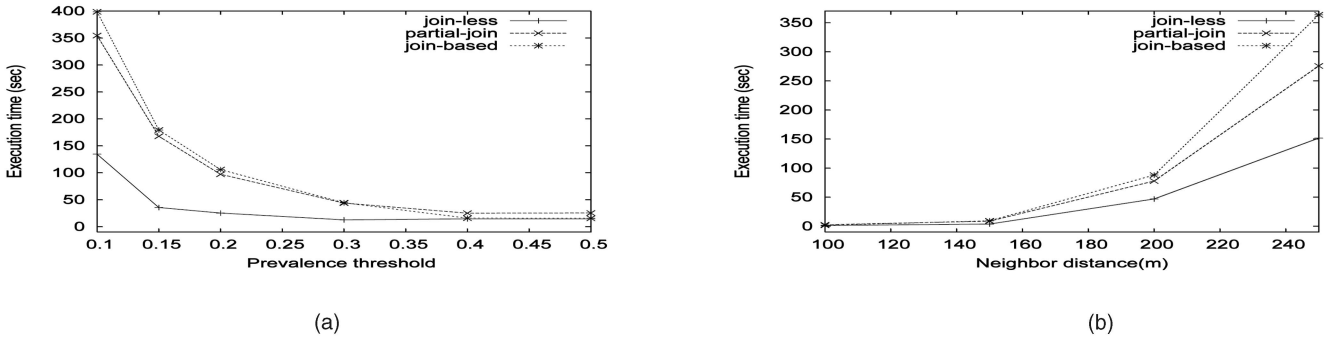


Fig. 11. Real data sets: (a) a climate data set and (b) a chimpanzee behavior data set.

of pruning the candidate colocations increased since the participation ratios of features of overlap points decrease.

7.2.2 Scalability of the Joinless Algorithm

We examined the scalability of the joinless algorithm with several workloads, i.e., different numbers of point objects, numbers of feature types, neighbor distance thresholds, and prevalence thresholds. We compared the total computation time of finding all prevalent colocations including the materialization cost.

Effect of the number of point objects. First, we compared the effect of the number of points in the different algorithms. We used two different spatial frame sizes, $10,000 \times 10,000$ and $1,000 \times 1,000$. In the first frame, even if the number of points is increased from 10K to 50K, the three algorithms showed similar execution time since the data sets are still sparse. However, in the second frame, with the increase of number of points, the join-based algorithm's execution time is dramatically increased due to the increase of data density. As shown in Fig. 9a, the joinless algorithm shows scalability to large dense data sets.

Effect of the number of features. In the second experiment, we compared the performance of the algorithms as a function of the number of different features. We also used two different dense data sets of 15K points. Fig. 9b shows the results. In the sparse data set, the algorithms show similar execution time even if the number of features increases. In the dense data set, overall execution time is decreased with the increase of features. The reason is that, under the same number of points, the increase of features causes the number of points per each feature to be decreased, which in turn may lead to a decrease in the number of instances per colocation. Overall, the joinless algorithm shows better performance.

Effect of neighbor distance. The third experiment examined the effect of different neighbor distances. As shown in Fig. 9c, the joinless algorithm shows less increase in the execution time with the increase of distance threshold. The join-based algorithm shows a rapid increase since the increase of neighbor distance makes the neighborhood areas larger and increases the number of colocation instances.

Effect of prevalence threshold. In the final experiment of scalability, we examined the performance effect as the prevalence threshold increases. Overall, the execution time decreases for all three algorithms with the increase of the prevalence threshold, as shown in Fig. 9d. However, the joinless algorithm reduces the computation time by a magnitude at lower threshold values.

7.2.3 Behavior of the Partial Join Algorithm

We examined the behavior of the partial join algorithm with the density of neighborhood areas and the cut instances by the explicit partition of spatial data.

Effect of data density in neighborhood areas. The density of neighborhood areas was controlled by $clumpy$ degree in Fig. 10a. The synthetic data sets had around 10 percent cut neighbor relations by disjoint clique partitioning in the partial join algorithm. The partial join algorithm performed well even if the data density increases with $clumpy$ degree. The reason is that the points are clustered in some neighborhood areas with fixed cut relations. By contrast, the joinless algorithm shows an increase in execution time with the increase of density. The join-based algorithm always shows a higher execution time than the other algorithms.

Effect of cut relations. In the second experiment, under a fixed clumpy degree of 15, we increased the number of cut relations. The number of cut relations is controlled by the $across$ ratio in synthetic data generation. At low $across$ ratios, the partial join algorithm with fewer cut relations shows better performance than the joinless algorithm. However, the execution time increases with the increase of the $across$ ratio because of the increase of interneighborhood instances and the increase of instance join operations to trace them. Overall, the join-based algorithms increase in execution time with an increase of the across ratio due to the increase of neighbor density.

7.2.4 Evaluation with Real Data Sets

We used two different types of real-world data sets. One was an Earth climate data set related to vegetation growth [16]. Another was an Ecology animal behavior data set that contained chimpanzee observation data from 1999 to 2001 [4].

1. *Earth climate data*. The earth climate data set includes monthly measurements of variables such as global plant growth, e.g., Net Primary Production (NPP), and climate variables, e.g., precipitation (PREC), on latitude-longitude spherical grids. An example of a discovered colocation pattern was (NPP-Hi, PREC-Low), where Hi(Low) denotes an unusually high(low) value of the measurements. The total number of event features was 18. The total number of instances was 15,515. We used 4 as a neighborhood distance, which means four cells (each grid cell is 1 degree $\times$ 1 degree). In Fig. 11a, the joinless method shows much better performance at the lower

threshold values. The performance difference between the partial join method and the join-based method is relatively small because the cut relation ratio was almost 0.8.

2. *Ecology animal behavior data*. The animal behavior data set has 24 chimpanzee features. We assigned a unique instance id to different location points per chimpanzee id. The total number of point instances was 698. Fig. 11b presents the execution time of the algorithms by different neighbor distances. The execution time of the joinless method increases more slowly than the other methods with the increase of distance and shows better performance. According to domain scientists in the application [4], neighbor distances between 200 and 300 are considered meaningful distance thresholds. In this context, our joinless method showed much better performance compared with the join-based method.

8 Conclusion and Future Work

In this paper, we propose the materialization of neighborhood information for efficient colocation pattern mining. We present two neighborhood materialization models, a star partition model and a clique partition model, with no loss of colocation instances. We compare the advantages and disadvantages of the two models in colocation pattern mining and present the instance filtering schemes and prevalent colocation filtering schemes. We develop a joinless colocation mining algorithm on the star partition model. The algorithm is efficient since it does not require expensive spatial joins or instance joins but uses an instance-lookup scheme for identifying colocation instances. We prove the joinless algorithm is correct and complete in finding colocation rules. In addition, we describe a partial join approach for spatial data which are clustered in neighborhood areas. We analyzed the computational complexity of the joinless algorithm and the partial join algorithm, and provided analytical comparisons with the previous instance-join-based algorithm. In the experimental evaluation using synthetic and real-world data sets, we examined the effects of our algorithm design decision, the scalability of the joinless algorithm and the behavior of the partial join algorithm. The join-based method appears to be appropriate for sparse data sets. However, the joinless method outperforms the join-based method in overall parameter settings and shows scalability to dense data.

Many application domains include spatio-temporal features. Scientists in these domains are interested in understanding the evolution of colocation patterns among stationary features as well as colocation patterns over moving feature types. In future work, we plan to explore methods to answer temporal questions such as how a colocation changes over time as well as methods to identify moving object colocation patterns.

Acknowledgments

This work was partially supported by US National Science Foundation grant 0431141 and Oak Ridge National Laboratory. The content of this work does not necessarily reflect the position or policy of the government and no official endorsement should be inferred. The authors are thankful to Kim Koffolt for her feedback that helped improve the readability of this paper.

References

[1] J. Roddick and M. Spiliopoulou, "A Bibliography of Temporal, Spatial and Spatio-Temporal Data Mining Research," *Proc. SIGKDD Explorations,* vol. 1, no. 1, pp. 34-38, 1999.

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

[3] S. Shekhar, P. Zhang, Y. Huang, and R. Vatsavai, "Trends in Spatial Data Mining," *Data Mining: Next Generation Challenges and Future Directions,* AAAI/MIT Press, 2004.

[4] "Nsf-Sei Project: Spatio-Temporal Analysis for Ecology Behavior," http://www.cs.umn.edu/research/chimps/, 2006.

[5] R. Agarwal and R. Srikant, "Fast Algorithms for Mining Association Rules," *Proc. 20th Conf. Very Large Databases,* 1994.

[6] K. Koperski and J. Han, "Discovery of Spatial Association Rules in Geographic Information Databases," *Proc. Fourth Int'l Symp. Large Spatial Data Bases,* pp. 47-66, 1995.

[7] S. Shekhar and Y. Huang, "Colocation Rules Mining: A Summary of Results," *Proc. Int'l Symp. Spatio and Temporal Database (SSTD),* 2001.

[8] Y. Morimoto, "Mining Frequent Neighboring Class Sets in Spatial Databases," *Proc. ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining,* 2001.

[9] D. Papadias, N. Mamoulis, and Y. Theodoridis, "Processing and Optimization of Multiway Spatial Joins Using R-Trees," *Proc. Symp. Principles of Database Systems,* 1999.

[10] H. Park, G. Cha, and C. Chung, "Multi-Way Spatial Joins Using R-Trees: Methodology and Performance Evaluation," *Proc. Sixth Int'l Symp. Advances in Spatial Databases,* 1999.

[11] J. Yoo and S. Shekhar, "A Partial Join Approach for Mining Colocation Patterns," *Proc. ACM Int'l Symp. Advances in Geographic Information Systems (ACM-GIS),* 2004.

[12] M. Kuramochi and G. Karypis, "Frequent Subgraph Discovery," *Proc. IEEE Int'l Conf. Data Mining,* 2001.

[13] X. Yan and J. Han, "gSpan: Graph-Based Substructure Pattern Mining," *Proc. IEEE Int'l Conf. Data Mining,* 2001.

[14] J. Yoo and S. Shekhar, "A Join-Less Approach for Co-Location Pattern Mining: A Summary of Results," *Proc. IEEE Int'l Conf. Data Mining (ICDM),* 2005.

[15] L. Arge, O. Procopiuc, S. Ramaswamy, T. Suel, and J. Vitter, "Scalable Sweeping-Based Spatial Join," *Proc. Int'l Conf. Very Large Databases,* 1998.

[16] "Discovery of Patterns in the Global Climate System Using Data Mining," http://www.ahpcrc.umn.edu/nasa-umn/, 2006.

Jin Soung Yoo received the BS degree in statistics and the BS degree in computer science and engineering from Korea University in 1992. She worked as a software engineer for Samsung SDS Inc. Seoul, Korea, from 1992 to 1999. She is currently working toward the PhD degree in computer science at the University of Minnesota, Minneapolis. Her research interests include databases, data mining, spatial/spatio-temporal data analysis, and network/graph mining. She is a student member of the IEEE, the IEEE Computer Society, and the ACM.

Shashi Shekhar received the BTech degree in computer science from the Indian Institute of Technology, Kanpur, India, in 1985, the MS degree in business administration, and the PhD degree in computer science from the University of California, Berkeley, in 1989. He is a McKnight Distinguished University Professor at the University of Minnesota, Minneapolis. His research interests include spatial databases, spatial data mining, geographic and information systems (GIS), and intelligent transportation systems. He has served on the editorial board of the *IEEE Transactions on Knowledge and Data Engineering* as well as the IEEE-CS Computer Science and Engineering Practice Board and is serving as a coeditor-in-chief of *Geoinformatica*. He is a fellow of the IEEE and a member of the ACM.