Colocation Mining: Exploring Local and Regional Interesting Patterns with the Map-Based Instance Table Approach

Abigail Kelly, Arpan Sainju
amk7r@mtmail.mtsu.edu,arpan.sainju@mtsu.edu
Middle Tennessee State University
Murfreesboro, Tennessee, USA

Abstract

A colocation pattern refers to spatial features that frequently appear near each other within a geographical area. Colocation pattern mining is crucial in identifying spatial patterns and relationships between objects or events, aiding decision-making in urban planning, environmental monitoring, and marketing. Existing work in colocation mining primarily addresses computational challenges like the exponential growth in candidate patterns and the computational expense of checking spatial neighborhood relationships for large feature instances. However, few studies address the challenge of varying interestingness of colocation patterns across different regions. Interesting colocation patterns are context-dependent and influenced by local contexts and cultural differences. For instance, a pattern that is interesting in Japan may not be interesting in the USA. Identifying the appropriate spatial neighborhood relationship constraint is complex. The general approach to colocation mining involves identifying colocated instances of patterns to determine their interestingness, which can be memory-intensive as the number of instances and candidate patterns grow. To address these challenges, we propose a novel approach that analyzes regional data to estimate the best spatial neighborhood relationship constraint. Our memory-optimized colocation mining framework uses approximately 70% less memory than existing implementations and can identify both sub-regional and regional colocation patterns. We evaluate our approach using the Global Terrorism Database, which includes terrorist attack events worldwide. Through case studies, we demonstrate that the spatial neighborhood relationship constraint and the interestingness of a pattern can vary significantly from region to region.

CCS Concepts: • **Information systems → Data mining**.

Keywords: Colocation Mining, Geospatial Data Mining

1 Introduction

Considering a collection of spatial features and their instances, colocation mining seeks to identify subsets of features whose instances are commonly found in close proximity. Examples of colocation patterns include symbiotic relationships between species such as Nile Crocodiles and Egyptian Plover [21], as well as interdependent businesses such as restaurants and gas stations. Due to spatial diversity, interesting colocation patterns may not be consistent across a geographical area. For instance, certain patterns may be considered interesting in one region, while entirely different patterns may be interesting in another region. If we consider the popularity of social media platforms in different regions, WhatsApp and Instagram rank as the top two most popular platforms based on user numbers in India, while LINE and Twitter take the lead in Japan [1].

Societal Applications: Colocation mining holds significance across various applications that seek to uncover connections among diverse spatial events or factors and may yield important insights for many applications. For example, in the realm of public safety, law enforcement agencies strive to discern correlations between various types of crime events and potential sources of criminal activity. In ecology, scientists examine shared spatial patterns among different species to capture insights into their interactions and spatial distributions. Within the domain of public health, the identification of links between human diseases and potential environmental triggers stands as a crucial challenge. In the field of climate science, colocation patterns play a pivotal role in unveiling relationships between the occurrences of distinct climate extreme events [21]. Additionally, when conducting business analysis for determining the optimal location of a new business, it's essential to understand the types of businesses in close proximity that could contribute to its success [26].

Challenges: There are several challenges in colocation mining. The colocation mining algorithm requires finding colocated instances of each pattern, which involves checking the spatial neighborhood relationships between instances of different features. This can be computationally expensive as the number of instances grows. Additionally, the number of candidate colocation patterns can grow exponentially with the number of features. Evaluating a large number of candidate patterns can be computationally expensive. Furthermore, storing the intermediate collection of colocated instances for each pattern can be memory intensive as the number of instances and candidate patterns grows. Finally, determining the best criteria to determine neighborhood

relationships is nontrivial and may vary from region to region. Additionally, the interesting patterns may differ from one region to another. Most of the existing work primarily focuses on addressing the computational challenges of the colocation mining problem [12, 20, 21, 28].

Contributions: We make the following contributions in the paper: (1) We design and implement a novel framework for mining regional colocation patterns, which includes a spatial neighborhood relationship constraint estimation algorithm and a colocation mining algorithm with a map-based instance table; (2) We provide theoretical analysis of the correctness and completeness of our approach; (3) We conduct experiments on both real-world data sets and synthetic data sets with varying parameters.

Scope and Outline: We focus on spatial colocation patterns that are defined by the event-centric models [12]. We also assume that the underlying space is defined in Euclidean space. Finally, we will primarily focus on addressing the challenge of determining the optimal spatial neighborhood relationship constraints for various regions and the storage of intermediate data in the colocation algorithm.

The outline of this paper is as follows. Section II analyzes related work. Section III goes over basic concepts and the definition of colocation spatial mining. Section IV introduces our proposed algorithm. Sections V and VI summarize our experimental evaluation of the proposed algorithm with results from synthetic data sets and case studies from a real-world data set. Section VII concludes the paper with potential future research directions.

2 Related Work

Colocation pattern mining has been studied extensively. Early work includes colocation patterns based on event-centric models [12] and spatial association rule mining [14, 16]. There are works on the partial-join algorithm [28] and the joinless algorithm [27], as well as the multi-resolution upper bound filter [12] and the grid-based approach [21]. Parallel colocation mining algorithms using GPU, which include refinement algorithms and memory optimizations [20, 21] are being studied. Spatiotemporal colocation mining [3, 19] and dynamic colocation mining [9] are being studied as well. Most of the existing works in colocation pattern mining have primarily focused on enhancing computational efficiency, typically requiring user-specified neighborhood relationship constraints and a prevalence threshold for pattern discovery.

Estimating Spatial Neighborhood Relationship Constraint: Most of the colocation mining algorithms assume a fixed distance threshold to determine neighborhood relationships [12, 13, 20, 21, 27], which is user-defined. Work on estimating the neighborhood relationship constraint includes using algorithms based on Delaunay triangulation [4, 7, 22, 23]. However, the accuracy of Delaunay triangulation techniques depends on the size of the data [23]. There are works that require the user to iteratively select edges to specify the neighborhood relationship constraint [18]. Other works employ a nearest neighbor-based approach to estimate the neighborhood relationship constraint [17, 26]. Nonetheless, users are still tasked with identifying the optimal 'k' value for the k-nearest neighbors (kNN) algorithm.

Addressing Memory Limitation: Most of the colocation mining algorithms have primarily focused on enhancing computational efficiency. However, some works have introduced different indexing structures to store neighborhood relationships for efficient retrieval. Examples include Zonal colocation mining [5], a Max-Min CP-Tree [23], CPI-Tree [24], and an iCPI-Tree approach [25]. Nevertheless, all of these approaches generate intermediate instance tables but do not optimize the storage in memory.

Identifying Regional and Sub-Regional Patterns: Regional colocation mining is another type of colocation mining, but it focuses on finding interesting patterns on different scales (regional vs. sub-regional). Work on finding regional associations between continuous variables [10] and statistically significant regional colocation patterns [4] is being conducted. Additionally, reward-based regional discovery algorithms [8], regional interest measures [15], and multi-level regional colocation frameworks that detect global patterns first then regional patterns [6] are being studied. Existing approaches may suffer from increased computational complexity, especially when prioritizing global pattern detection over regional patterns, potentially limiting scalability.

In this paper, we propose a novel algorithm that dynamically estimates the region-specific spatial neighborhood relationship constraint without extra parameter requirements. In addition, we use a memory-efficient map-based approach to reduce intermediate memory usage. We find both sub-regional and regional colocation patterns.

3 Problem Statement

This section describes the basic concepts of colocation mining and the problem definition with an example in Fig. 1.

3.1 Basic Concepts

Region: A *region* is an area of interest, often comprising countries that share borders. These individual countries can be considered ***sub-regions*** within the larger region. For example, in Fig. 1, the region is the space comprising both Sub-Region 1 and Sub-Region 2.

Spatial Feature: A *spatial feature* is a categorical attribute such as a terrorist attack type (e.g., assassination, bombing, hijack). For example, in Fig. 1, there are four spatial features A, B, C, D.

Feature Instance: A *feature instance* is the occurrence of a spatial feature at the same or different geographical location (e.g., multiple instances of the terrorist attack type

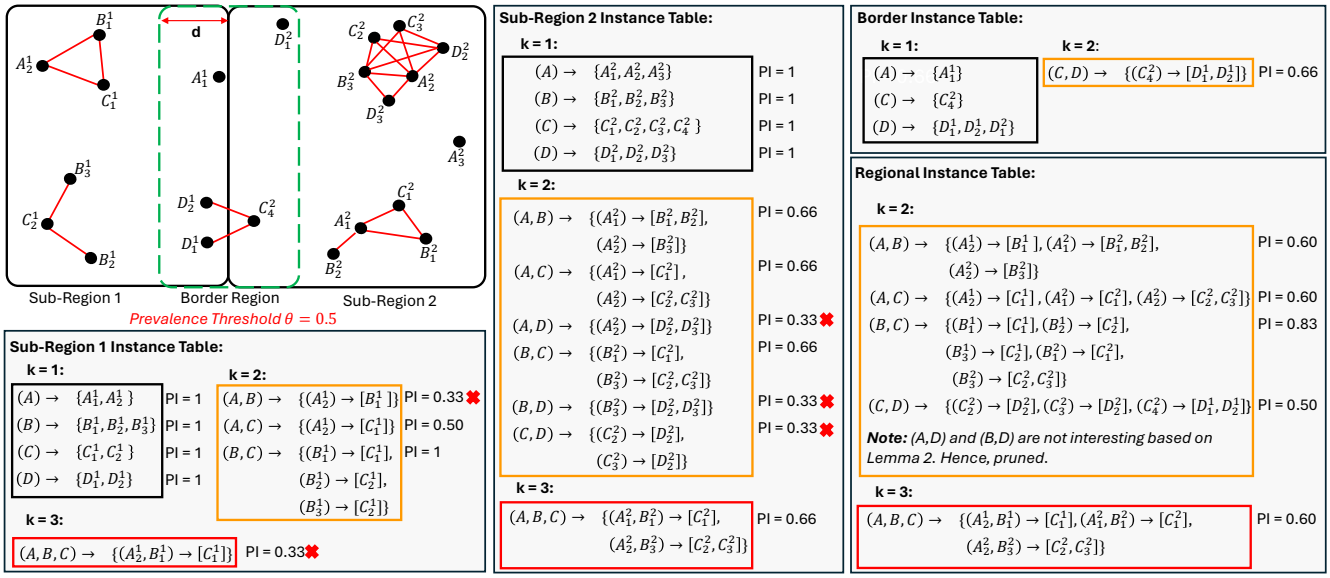


Figure 1. Problem Example with Map-Based Instance Table

"hijacking"). For example, there are three instances of B in Sub-Region 2: B^2_1, B^2_2, B^2_3.

Distance Threshold: The *distance threshold* is the spatial neighborhood relationship constraint. Any two instances of different features are considered to be in a neighborhood relationship if and only if the distance between them is less than or equal to the distance threshold. In this paper, we will be using the terms "spatial neighborhood relationship constraint" and "distance threshold" interchangeably.

Spatial Neighbors: *Spatial neighbors* are two feature instances that are separated by a distance that is smaller than or equal to a distance threshold.

Clique: A *clique* is two or more instances if every pair of instances are spatial neighbors.

Colocation Pattern Instance: A *colocation pattern instance* is when a set of instances in a clique are of different feature types. For example, in Sub-Region 2, $\{A^2_1, B^2_1, C^2_1\}$ is an instance of the colocation pattern (A, B, C).

Colocation Pattern: A *colocation pattern* is a corresponding set of features to the colocation pattern instance.

Cardinality: The *cardinality* is the size of a set. For example, the cardinality of the colocation pattern (A, B, C) is 3 because there are 3 features in the pattern.

Instance Table: An *instance table* is a map-like structure with key-value pairs. The keys are instances of a subset of features of a pattern, and the value is the instances of a subset of features that are co-located with the instance of the key. For example, in Sub-Region 2, the instance table of the colocation pattern (B, C) has keys B^2_1 and B^2_3. The co-located instance of B^2_1 is C^2_1, and the co-located instances of B^2_3 are C^2_2 and C^2_3. A ***row instance*** (RI) of the instance table is a pair consisting of the key and one of its values, denoted $RI(P)$ where P is a colocation pattern. For example, in Sub-Region 2, a row instance of (B, C), $RI((B, C))$, is $\{(B^2_1) \rightarrow [C^2_1]\}$ where B^2_1 is the key and C^2_1 is the value. A ***table instance*** (TI) of the instance table is the collection of all the row instances of a colocation pattern, denoted $TI(P)$ or $TI(P, f)$ where f is a feature of colocation pattern P. For example, the table instance of (B, C), $TI((B, C))$, is $\{(B^2_1) \rightarrow [C^2_1], (B^2_3) \rightarrow [C^2_2, C^2_3]\}$, and the table instance of the feature B, $TI((B, C), B)$, of the colocation pattern (B, C) is $\{B^2_1, B^2_3\}$.

Candidate Colocation Pattern: A *candidate colocation pattern* refers to colocation patterns whose participation index is undecided.

Participation Ratio (PR): A *participation ratio* of a spatial feature within a candidate colocation pattern is the ratio of the number of unique feature instances that participate in colocation instances to the total number of feature instances. For example, in Sub-Region 2, the participation ratio of A the candidate colocation pattern (A, B) is $\frac{2}{3}$ since two instances of A: A^2_1 and A^2_2 participate in the colocation pattern instances $\{(A^2_1) \rightarrow [B^2_1, B^2_2], (A^2_2) \rightarrow [B^2_3]\}$.

Participation Index (PI): The *participation index* of a candidate colocation pattern is the minimum participation ratio among all the member features. For example, in Sub-Region 2, the participation ratio of the candidate colocation pattern (A, B) is $\frac{2}{3}$ because $\min(\frac{2}{3}, 1) = \frac{2}{3}$.

Prevalence Threshold: The *prevalence threshold* is a user-defined minimum threshold for participation index.

Prevalent: A candidate colocation pattern is *prevalent* if and only if its PI is greater than or equal to the prevalence threshold. It signifies the pattern's feature instances are frequently

located together in the same neighborhood cliques and the pattern is interesting.

3.2 Problem Definition

Given:

- A set of spatial features and their instances
- Prevalence threshold: θ

Find:

- Spatial neighborhood relationship constraint: d
- All colocation patterns with $PI \geq \theta$ in sub-regions and entire region

Objective:

- Estimate the spatial neighborhood distance threshold
- Reduce memory utilization

Fig. 1 provides a problem example. The input data contains 22 instances of 4 spatial features for both Sub-Region 1 and Sub-Region 2. The neighborhood relationship constraint is d and the prevalence threshold is $\theta = 0.5$. The border region includes instances that lie within the distance threshold from the shared border between Sub-Region 1 and Sub-Region 2. The prevalent patterns in Sub-Region 1 are (A, C) with $PI = 0.50$ and (B, C) with $PI = 1$. The prevalent patterns in Sub-Region 2 are (A, B) with $PI = 0.66$, (A, C) with $PI = 0.66$, (B, C) with $PI = 0.66$, and (A, B, C) with $PI = 0.66$. The prevalent pattern in the Border Region is (C, D) with $PI = 0.66$. The Sub-Region 1, Sub-Region 2, and Border Region's instance tables of cardinality $k = 2$ are merged to produce the regional instance table of $k = 2$, which is then used to determine the regional instance tables of higher cardinality. The prevalent patterns in the entire region are (A, B) with $PI = 0.60$, (A, C) with $PI = 0.60$, (B, C) with $PI = 0.83$, (C, D) with $PI = 0.50$, and (A, B, C) with $PI = 0.60$. The above patterns are prevalent because their $PI \geq \theta$.

4 Proposed Approach

This section introduces our novel colocation mining algorithm, which encompasses the estimation of spatial neighborhood relationship constraints and the identification of both sub-regional and regional colocation patterns. We begin with an overview of the algorithm's structure. Then, we describe the process of estimating spatial neighborhood relationship constraints, followed by implementation details of our algorithm for identifying both sub-regional and regional colocation patterns.

4.1 Algorithm Overview

The overall structure of our algorithm can be seen in Algorithm 1. The algorithm begins by estimating the spatial neighborhood relationship constraint in step 1. This neighborhood relationship constraint will later be used to calculate the prevalent patterns in the region and its sub-regions. The algorithm identifies the prevalent colocation patterns iteratively in sub-regions and then for the entire region.

Algorithm 1 Regional Colocation Miner

Input: Sub-regions S
Input: A set of spatial features for each sub-region F
Input: Instances of each spatial feature $I[F]$
Input: Minimum prevalence threshold θ
Output: All prevalent sub-regional (s) and regional (R) colocation patterns P^s and P^R respectively

1: $d \leftarrow$ DYNAMIC_DISTANCE_ESTIMATE(F)
2: **for** s **in** S **do**
3: Initialize $k \leftarrow 1$
4: Initialize $C^s_k \leftarrow F$, $P^s_k \leftarrow F$
5: Initialize $C^s_{k+1} \leftarrow$ APRIORI_GEN($P^s_k, k+1$)
6: Initialize instance tables I^s_k with $I[F]$ in region s
7: Initialize $I^s_{k+1} \leftarrow \emptyset$, $P^s_{k+1} \leftarrow \emptyset$
8: $N^s \leftarrow$ CALCULATE_STAR_NEIGHBORS(s)
9: **while** $|C^s_{k+1}| > 0$ **do**
10: $(P^s_{k+1}, I^s_{k+1}) \leftarrow$ CALC_PATTERNS($k+1, I^s_k$)
11: $P^s \leftarrow P^s \cup P^s_{k+1}$
12: $k = k + 1$
13: $C^s_{k+1} \leftarrow$ APRIORI_GEN($P^s_k, k+1$)
14: **end while**
15: **end for**
16: $k \leftarrow 2$
17: $I^b_k \leftarrow$ CALCULATE_BORDER_PATTERNS(k)
18: $I^R_k \leftarrow$ COMBINE_INSTANCE_TABLES(I^S_k, I^b_k)
19: $P^R_k \leftarrow$ PI(I^R_k)
20: $C^R_{k+1} \leftarrow$ APRIORI_GEN($P^R_k, k+1$)
21: **while** $|C^R_{k+1}| > 0$ **do**
22: $(P^R_{k+1}, I^R_{k+1}) \leftarrow$ CALC_PATTERNS($k+1, I^R_k$)
23: $P^R \leftarrow P^R \cup P^R_{k+1}$
24: $k = k + 1$
25: $C^R_{k+1} \leftarrow$ APRIORI_GEN($P^R_k, k+1$)
26: **end while**
27: **return** P^S, P^R

In Algorithm 1, steps 3 to 14 are performed for each sub-region s. The candidate colocation patterns and their instance tables of cardinality $k+1$ are generated based on prevalent patterns and their instance tables of cardinality k. Each candidate pattern of cardinality $k+1$ is then evaluated based on the participation index computed from its instance table. For cardinality $k = 1$, prevalent colocation patterns simply consist of the set of input features, and their instance tables are the instance lists for each feature (steps 3 and 4). Step 5 generates candidate patterns C^s_{k+1} of cardinality $k+1$ in sub-region s based on the prevalent patterns P^s_k of the same region using Apriori property [12]. Based on the Apriori property, a candidate pattern of size $k+1$ cannot be prevalent if any of the subset patterns of size k are not prevalent. Step 8 calculates the star neighbors of each instance in the sub-region s. The star neighbors are generated by using an R-tree. We first build the R-tree by inserting all instances. For

each instance, we perform a spatial query on the R-tree to find instances of other feature types (excluding the current instance's feature type). Next, we filter the results to exclude the current instance and any others that do not meet the spatial neighborhood criteria. The remaining instances form the star neighborhood for the given instance. Steps 10 to 13 are repeated until there are no more candidate patterns. In step 10, the instance tables (I^s_{k+1}) for each candidate pattern of size $k+1$ in sub-region s are generated and the prevalent colocation patterns P^s_{k+1} are identified (more details on this will be provided later). In step 11, the set P^s that represents all the prevalent colocation patterns in sub-region s is updated by taking the union of P^s with the prevalent colocation patterns P^s_{k+1}. Step 12 updates the cardinality size k by 1 and step 13 generates the candidate colocation patterns for cardinality size $k+1$ using Apriori property on the prevalent colocation patterns of size k (i.e. P^s_k). After the prevalent patterns are calculated for all the sub-regions, we set the cardinality size k to 2 in step 16 and then calculate the border region instances and patterns in step 17. The border region instances are determined by creating a buffer region circle with a radius equal to the spatial neighborhood relationship constraint around each instance. If the buffer circle of an instance intersects a border of the region, that instance is marked as a border region instance. Once all the border region instances are determined, the instance tables I^b_k of candidate colocation patterns of cardinality $k=2$ for the border region b are calculated. I^b_k consists only of instances of patterns with feature instances that occur in 2 different sub-regions. For example, in Fig. 1, the instance $\{C^2_4, D^1_1\}$ of colocation pattern (C, D) would appear in I^b_k because C^2_4 occurs in Sub-Region 1 and D^1_1 occurs in Sub-Region 2. Step 18 merges the instance tables of respective candidate colocation patterns from all sub-regions S with the instance table of the border region. This unified table represents the instance table of the candidate colocation patterns across the entire region R and is denoted as I^R_k. Step 19 computes the participation index for each size $k=2$ pattern across the entire region. The resulting prevalent colocation patterns, P^R_k, is then utilized in step 20 to generate candidate patterns for size $k+1$ employing the Apriori property. Finally, we replicate the process similar to steps 9-14 from step 21 to step 26 to identify prevalent colocation patterns across the entire region. Fig. 1 shows the execution trace for a region with two sub-regions for cardinality size $k=1$, $k=2$ and $k=3$.

Lemma 1. *Let R be a region with n sub-regions $s_1, s_2, \ldots, s_n$. Let C be a colocation pattern such that $PI(C) \geq \theta\ \forall s_1, s_2, \ldots s_n$ where θ is the prevalence threshold. Then, $PI(C) \geq \theta$ for R.*

Proof. Let f_i be a feature of cardinality k colocation pattern $C = (f_1, f_2, \ldots, f_k)$.
Denote $I^s = \{I^{s_1}_{f_i}, \ldots, I^{s_n}_{f_i}\}$ as a set of the instances of feature f_i participating in C in sub-regions $s_1, \ldots, s_n$.
The participation ratio (PR) of each f_i of C in each sub-region is denoted $PR^{s_p}(C, f_i) = \frac{|TI^{s_p}(C, f_i^{s_p})|}{|I^{s_p}_{f_i}|}\ \forall p \leq n$.
We know $\frac{|TI^{s_p}(C, f_i^{s_p})|}{|I^{s_p}_{f_i}|} \geq \theta$, so $\frac{\sum_{p=1}^{n} |TI^{s_p}(C, f_i^{s_p})|}{\sum_{p=1}^{n} |I^{s_p}_{f_i}|} \geq \theta$.
So, $PR^R(C, f_i^R) = \frac{\sum_{p=1}^{n} |TI^{s_p}(C, f_i^{s_p})|}{\sum_{p=1}^{n} |I^{s_p}_{f_i}|} \geq \theta$. Therefore, $PI^R(C) = \min(PR^R(C, f_1^R), PR^R(C, f_2^R), \ldots, PR^R(C, f_k^R)) \geq \theta$, making C a prevalent pattern for the entire region R. □

Lemma 2. *Let R be a region with n sub-regions $s_1, s_2, \ldots, s_n$, and m border regions $b_1, b_2, \ldots b_m$. A border region is an overlapping geographical area where two sub-regions touch. Let C be a colocation pattern and f be the feature in C such that $PR(C, f) < \theta\ \forall s_1, s_2, \ldots s_n$ and $PR(C, f) < \theta\ \forall b_1, b_2, \ldots b_m$. Then, $PI(C) < \theta$ for R.*

Proof. Denote $I^s_f = \{I^{s_1}_f, \ldots, I^{s_n}_f\}$ as a set of the instances of feature f participating in C in sub-regions $s_1, \ldots, s_n$.
Denote $I^b_f = \{I^{b_1}_f, \ldots, I^{b_m}_f\}$ as a set of the instances of feature f participating in C in border regions $b_1, \ldots, b_m$ where $I^{b_j}_f$ denotes the set of instances of f where at least two instances in the row instance of C occur in two distinct sub-regions.
The participation ratio (PR) of f in C for each sub-region and border region is denoted
$PR^{s_p}(C, f) = \frac{|TI^{s_p}(C, f)|}{|I^{s_p}_f|}\ \forall p \leq n$ and
$PR^{b_j}(C, f) = \frac{|TI^{b_j}(C, f)|}{|I^{b_j}_f|}\ \forall j \leq m$ respectively.
We know $\frac{|TI^{s_p}(C, f)|}{|I^{s_p}_f|} < \theta$ and $\frac{|TI^{b_j}(C, f)|}{|I^{b_j}_f|} < \theta$
for each sub-region and border region.
So, $|TI^{s_p}(C, f)| < \theta |I^{s_p}_f|$ and $|TI^{b_j}(C, f)| < \theta |I^{b_j}_f|$.
This implies $\sum_{p=1}^{n} |TI^{s_p}(C, f)| < \theta \sum_{p=1}^{n} |I^{s_p}_f|$ and
$\sum_{j=1}^{m} |TI^{b_j}(C, f)| < \theta \sum_{j=1}^{m} |I^{b_j}_f|$.
So, $\sum_{p=1}^{n} |TI^{s_p}(C, f)| + \sum_{j=1}^{m} |TI^{b_j}(C, f)|$
$< \theta(\sum_{p=1}^{n} |I^{s_p}_f| + \sum_{j=1}^{m} |I^{b_j}_f|)$,
which further implies
$PR^R(C, f) = \frac{\sum_{p=1}^{n} |TI^{s_p}(C, f)| + \sum_{j=1}^{m} |TI^{b_j}(C, f)|}{\sum_{p=1}^{n} |I^{s_p}_f| + \sum_{j=1}^{m} |I^{b_j}_f|} < \theta$.
Therefore, $PI^R(C) < \theta$,
making C not a prevalent pattern for the entire region R. □

Theorem 1. *The regional colocation pattern calculation framework is correct and complete.*

Proof. The algorithm is complete and does not mistakenly prune out any prevalent patterns due to Lemma 2. The algorithm is correct since it computes the exact participation index of each candidate pattern in each sub-region and the entire region. □

Step 18 in Algorithm 1 utilizes Lemma 1 and 2 to streamline computation by eliminating redundancy while processing candidate patterns for the entire region.

4.2 Dynamic Neighborhood Relationship Constraint

Algorithm 2 Dynamic Neighborhood Relationship Estimate

Input: A set of spatial features F
Input: Instances of each spatial feature $I[F]$
Output: Neighborhood relationship constraint d

1: Initialize $D \leftarrow \emptyset$, $K_{max} \leftarrow \sqrt{|I[F]|} + 1$, $A \leftarrow \emptyset$
2: Initialize memoization table $T \leftarrow \emptyset$ of size $|I[F]| \times K_{max}$
3: **for** f **in** F **do**
4: $S_f \leftarrow$ EXTRACT_BY_FEATURE_TYPE(f)
5: $I[F]_{Excludingfeature} \leftarrow I[F] - S_f$
6: $r \leftarrow$ RTREE()
7: ADD_POINTS_TO_RTREE($r, I[F]_{Excludingfeature}$)
8: **for** p in S_f **do**
9: $x \leftarrow p[0], y \leftarrow p[1]$
10: $N = r$.nearest($(x, y), K_{max}$)
11: D.append(SORT_NEIGHBORS_DISTANCES(N))
12: **end for**
13: **end for**
14: $T[:, 2]$ = ROWWISE_SUM($D[:, :3]$)
15: **for** i **in** range($|I[F]|$ **do**
16: **for** k **in** range(3, K_{max}) **do**
17: $T[i, k] = T[i, k-1] + D[i][k]$
18: **end for**
19: **end for**
20: C = COLUMNWISE_SUM(T)
21: **for** k **in** range(2, K_{max}) **do**
22: A.append($C[k]/(|I[F]| \times (k+1))$)
23: **end for**
24: $d \leftarrow$ KNEE_METHOD(A)
25: **return** d

The "Dynamic Neighborhood Relationship Estimate" algorithm is shown in Algorithm 2 and is designed to estimate the spatial neighborhood relationship constraint d for a given set of spatial features F and their instances $I[F]$. Step 1 sets the max number of neighbors value K_{max} to the square root of the total number of instances $|I[F]|+1$ in the region based on [11]. We initialize D and A as empty lists in step 1. D will eventually store the calculated K_{max} neighbors sorted by distances for each spatial feature instance and A will be used to store the average neighbor distances for varying values of k that range from 2 to K_{max}. Step 2 initializes a dynamic programming memoization table T with dimensions $|I[F]| \times K_{max}$ by setting all entries to empty. The algorithm then proceeds to process each feature in the set F. Steps 4 to 12 are performed for each feature $f \in F$. In step 4, the instances S_f of the feature type f are extracted. Step 5 involves creating a subset $I[F]_{Excludingfeature}$ by excluding instances of the current feature from $I[F]$. In step 6, a spatial index r using an R-Tree is created for $I[F]_{Excludingfeature}$, and step 7 adds the instances from this subset to the R-Tree for efficient spatial querying. For each instance p in S_f, steps 9 to 11 are executed. Step 9 extracts the x and y coordinates of p. In step 10, the algorithm finds the nearest neighbors N to the instance at (x, y) using the R-Tree. In Step 11, the neighbors are sorted by distances in ascending order. The sorted result is then appended to the list D. This process is repeated for all instances in S_f. After processing all features, the algorithm updates the memoization table T by summing the values of the first three columns of the list D, element-wise, and assigning the result to the third column of T (represented by index 2). Steps 15 to 19 involve iterating through each instance id i in $I[F]$ and updating T using dynamic programming. For each instance, step 16 iterates through possible values of k in the range 3 to K_{max} and updates $T[i, k]$ by adding the distance values from D. In step 20, the algorithm calculates the sum of values along each column of the memoization table T and assigns the result to the list C. In Steps 21 to 23, the algorithm iterates over the potential values of k within the range from 2 to K_{max}, updating the list A with information from C. This information is normalized by the number of instances and $k+1$ to accommodate the summation of $k+1$ neighbor distances. The resulting list A holds the average distance threshold for each k value. Finally, in step 24, the algorithm applies the "Knee Method" [2] to determine the optimal neighborhood relationship constraint d from the list A. By applying the "Knee Method", we identify the region of greatest curvature within the average distances. This is done by creating a straight line that connects the endpoints of the curve. The point on the curve where the difference between the curve and the straight line is the greatest is where the knee lies. Consequently, the peak of this curvature (knee) denotes the optimal k-value and the corresponding average distance is considered to be the spatial neighborhood relationship constraint.

4.3 Identifying Patterns using Map-based Instance Table

The "Map-Based Instance Table Pattern Calculation" algorithm is shown in Algorithm 3. This algorithm is designed to identify prevalent patterns P_k of size k from a set of candidate patterns C_k and their instances. In step 1, P_k is initialized as an empty set, I_k is initialized as an empty hash map to hold instances of candidate patterns, and H is initialized as another hash map to store unique instances of each feature in a candidate pattern in order to compute the participation ratios and participation index. Steps 2 to 35 iterate over each candidate pattern c in C_k. For each candidate pattern c, the algorithm extracts the base pattern $B_{pattern}$ from the first $k-1$ elements of c and the last feature L_f as the k-th (last) element of c in step 3. For example, the base pattern $B_{pattern}$ of degree 4 candidate pattern (A, B, C, D) is (A, B, C), and

Algorithm 3 Map-Based Instance Table Pattern Calculation

Input: List of candidate patterns C_k of size k
Input: Instance table I_{k-1} for all size $k-1$ patterns
Input: Hash Map that holds the starting and ending indices and instance count of each feature F_{info}
Input: Star neighbors of instances of each spatial feature S
Output: Filled in instance table I_k
Output: List of prevalent patterns P_k

1: Initialize $P_k \leftarrow \emptyset, I_k \leftarrow \emptyset, H \leftarrow \emptyset$
2: **for** c in C_k **do**
3: $B_{pattern} \leftarrow c[0:k-1], L_f \leftarrow c[k-1]$
4: Initialize $I_k[c] \leftarrow \emptyset, H[c] \leftarrow \emptyset$
5: $H[c] \leftarrow \{f : \emptyset$ **for** f **in** $c\}$
6: $I_{base} \leftarrow I_{k-1}[B_{pattern}]$
7: **for** key **in** I_{base} **do**
8: $N \leftarrow \emptyset$
9: **for** id **in** key **do**
10: **if not** N **then**
11: $N \leftarrow$ NEIGHBOR$(S[id], F_{info}[L_f])$
12: **else**
13: $N \leftarrow N \cap$ NEIGHBOR$(S[id], F_{info}[L_f])$
14: **end if**
15: **end for**
16: **for** i in $I_{base}[key]$ **do**
17: $n = N \cap$ NEIGHBOR$(S[i], F_{info}[L_f])$
18: **if** n **then**
19: $key_{new} = key$.append(i)
20: $I_k[c][key_{new}] \leftarrow n$
21: **for** $j \in key_{new}$ **do**
22: $f \leftarrow$ GET_FEATURE_ID(j)
23: $H[c][f]$.add(j)
24: **end for**
25: $H[c][L_f]$.update(n)
26: **end if**
27: **end for**
28: **end for**
29: $PR \leftarrow \emptyset$
30: **for** f **in** c **do**
31: PR.append$(|H[c][f]|/F_{info}[f].count)$
32: **end for**
33: $PI = \min(PR)$
34: P_k.append(c) **if** $PI \geq \theta$
35: **end for**
36: **return** P_k, I_k

the last feature L_f is D. Step 4 initializes c as a key in both I_k and H, with corresponding values initialized as empty hash maps. In step 5, for each feature f in the candidate pattern c, an entry in $H[c]$ is initialized with f as the key and an empty set as the value. In step 6, the instance table corresponding to the $k-1$ degree of $B_{pattern}$ is extracted and stored in I_{base}. Steps 8 to 27 are repeated for each key in I_{base}. Step 8 initializes an empty set N to hold the instances of feature type L_f that are the common neighbors to each instance id in key where each id in key represents an instance of a feature $f \in B_{pattern}$. Steps 10 and 11 find the neighbor of instance id of feature type L_f and update the list N if currently empty. The NEIGHBOR function checks the star neighbors of the current instance $S[id]$ and performs a binary search to identify the neighbors of type L_f using the F_{info} data structure, which holds the starting and ending indices and instance count of each feature (the instances are pre-sorted by feature type, ensuring that all instances of the same feature type are contiguous). In steps 12 and 13, if N is not empty, we perform a set intersection operation on N with the common neighbors of type L_f based on the remaining ids. This ensures that N contains only neighbors of type L_f that are common among the instances in the current key in I_{base}. For example, assuming c is (A, B, C, D), based on Fig. 1 regional instance table, L_f is D, I_{base} is (A, B, C) and its $keys$ are (A_2^1, B_1^1), (A_1^2, B_1^2), and (A_2^2, B_3^2). N for the key (A_2^2, B_3^2) is $[D_2^2, D_3^2]$. In steps 16 to 27, for each i in the value list of current key in I_{base}, we find its neighbors of type L_f and apply intersection operation with N; the result is then stored in n. For the key (A_2^2, B_3^2) the value list representing the common neighbor of type C for A_2^2 and B_3^2 is $[C_2^2, C_3^2]$ and n for both C_2^2 and C_2^3 is $[D_2^2]$. In steps 18 and 19, if n is not empty, we construct a new key key_{new} for the instance table of the current candidate pattern c by appending the current i. For the key (A_2^2, B_3^2) the new keys, key_{new}, are (A_2^2, B_3^2, C_2^2) and (A_2^2, B_3^2, C_3^2). Step 20 assigns the set of common neighbors n to the new key, key_{new}, in the instance table $I_k[c]$. Hence for (A, B, C, D), $I_k[(A, B, C, D))] \rightarrow \{(A_2^2, B_3^2, C_2^2) \rightarrow [D_2^2], (A_2^2, B_3^2, C_3^2) \rightarrow [D_2^2]\}$. In steps 21 to 24, the algorithm iterates over each instance j in the newly formed key, key_{new}. For each instance, it retrieves the feature ID f using the GET_FEATURE_ID function and adds j to the corresponding set $H[c][f]$—ensuring the storage of only one copy of each unique instance. Upon completion of processing all instances within key_{new}, the algorithm proceeds to update the set $H[c][L_f]$ with the instances contained in n. In steps 29 to 35, the algorithm uses the hash map $H[c]$ and F_{info} to compute the Participation Ratios (PR) and Participation Index (PI), which is the minimum of the PRs. If the PI of the current pattern c is greater than or equal to the prevalence threshold θ, then the pattern c is added to the list of prevalent patterns P_k in step 34. Finally, after all the candidate patterns are processed, the list of prevalent patterns P_k and their instance tables I_k are returned.

5 Evaluation

The goals of our evaluation are to:

- Evaluate the difference in spatial neighborhood relationship constraints across 3 case study regions.
- Compare the memory usage of our proposed map-based approach with array-based approaches

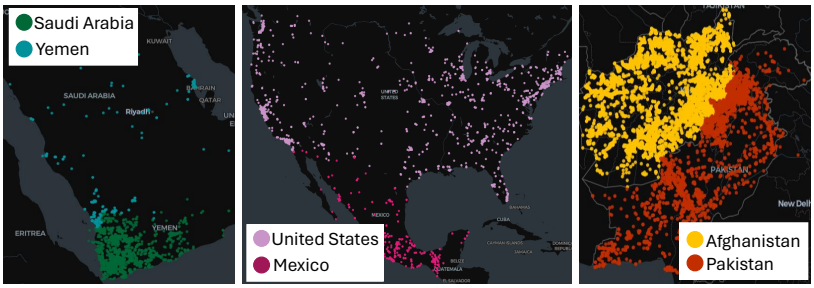
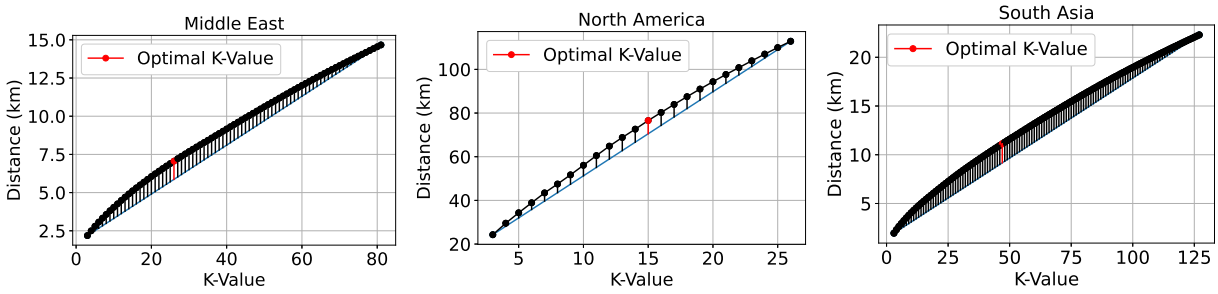


Figure 2. Case Study Regions of Interest



(a) (b) (c)

Figure 3. K-value vs. Distance graphs (a) Middle East (b) North America (c) South Asia

5.1 Experimental Setup

We implemented an algorithm that estimates the spatial neighborhood relationship constraint and calculates the important patterns in sub-regions and the entire region with a map-based instance table approach. The spatial neighborhood relationship constraint estimation was implemented in Python and the map-based colocation algorithm was implemented in C++ and run on a Dell workstation with Intel(R) Xeon(R) Gold 6248R CPU @ 3.00GHz with 256 GB main memory. The array-based results used for comparison with our approach are based on the findings of [12, 20, 21]. Since the array-based baseline approaches are correct and complete, they will use the same amount of memory when storing the intermediate instance tables.

5.2 Data Set Description

The real-world data set contains 8 attack types, which can be seen in Table 1. This data set is based on the Global Terrorism Database, which contains 215k terrorist attack event instances that occurred around the world between the years 1970-2020 and can be downloaded from https://www.start.umd.edu/gtd/contact/download. From the real-world data set, three regions were tested, as shown in Fig. 2. The case studies were conducted on subsets of this data with a variety of data densities.

Table 1. Attack Types

Attack Type	Identifier
Armed Assault	0
Assassination	1
Bombing	2
Facility Attack	3
Hijacking	4
Hostage Taking (Barricade)	5
Hostage Taking (Kidnapping)	6
Unarmed Assault	7

The synthetic data was generated similarly to [12]. We chose a study area of the size 10000×10000, a spatial neighborhood relationship constraint of 10, a prevalence threshold of $\theta = 0.5$, a maximal pattern cardinality of 5, and the number of maximal colocation patterns as 2. The total number of features is set as 12 and accounts for 2 additional noise features. While generating the number of instances for each maximal colocation pattern, their locations are distributed randomly according to the clumpiness, which is the number of overlaying colocation instances within the same neighborhood. High clumpiness implies large instance tables.

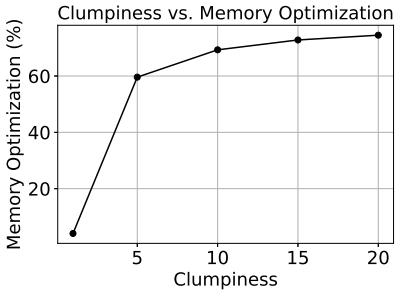


Figure 4. Memory Optimization with Varying Clumpiness

5.3 Results on Synthetic Data

In this experiment, five different clumpiness settings were tested: 1, 5, 10, 15, and 20 on a data set with 250k instances. As the clumpiness increases, the number of neighbors increases, resulting in a larger instance table. We can observe that as the clumpiness increases, the memory optimization percentage increases, as shown in Fig. 4.

5.4 Results on Real World Data

We conducted experiments on three regions, all containing two sub-regions (countries) each as shown in Fig. 2. The first region (Case Study 1), has has a mix of both dense and sparse data, the second region (Case Study 2), has sparse data, and the third region (Case Study 3) has dense data. The choice of the prevalence threshold value is domain-specific. In this study, we set the prevalence threshold value $\theta = 0.55$. Analysis of the resulting interesting patterns can be carried out by domain experts such as the police department or the Federal Bureau of Investigation. Colocation patterns based on terrorist attacks may reveal insights into the operational strategies and network dynamics of terrorist organizations. By analyzing these patterns, security experts can identify vulnerable regions, anticipate future threats, and develop targeted counterterrorism measures to mitigate risks effectively.

Case Study 1: This case study on the Middle East includes sub-regions Saudi Arabia (709 instances) and Yemen (6,478 instances). The spatial neighborhood relationship constraint is 7.06 km as shown in Fig. 3(a). Our novel map-based approach uses 73.2% less memory than the existing array-based approach, as shown in Table 2. Interesting patterns for sub-regions and the entire region are shown in Table 3.

Case Study 2: This case study on North America includes sub-regions United States (2,913 instances) and Mexico (639 instances). The spatial neighborhood constraint is 76.56 km as shown in Fig. 3(b). Our novel map-based approach uses 65.5% less memory than the existing array-based approach, as shown in Table 2. Interesting patterns for the sub-regions and the entire region are shown in Table 3.

Case Study 3: This case study on South Asia includes sub-regions Pakistan (15,753 instances) and Afghanistan (21,718 instances). The spatial neighborhood constraint is 11.12 km as shown in Fig. 3(c). Our novel map-based approach uses 73.6% less memory than the existing array-based approach, as shown in Table 2. Interesting patterns for the sub-regions and the entire region are shown in Table 3.

6 Discussion

The spatial neighborhood relationship constraint estimated for the Middle East in Case Study 1 is 7.06 km, which is smaller than the spatial neighborhood relationship constraint estimated for the regions in the other two case studies. This is because most instances in this case study are from Yemen, which has very dense data and is not spread out much geographically. The spatial neighborhood relationship constraint estimated for North America in Case Study 2 is 76.56 km, which is larger than the spatial neighborhood relationship constraint estimated for the regions in the other two case studies. The instances in North America are very sparse when compared to the other two regions, resulting in a larger spatial neighborhood relationship constraint. The spatial neighborhood relationship constraint estimated for South Asia in Case Study 3 is 11.12 km, which is greater than the spatial neighborhood relationship constraint estimated in Case Study 1 and less than the neighborhood relationship constraint estimated in Case Study 2. The instances in South Asia are more dense than Case Study 1 and Case Study 2 but are spread out over a larger area than Case Study 1, resulting in a spatial neighborhood relationship constraint that is between the estimations for the other two regions.

In the results from the synthetic data, we can see that as the clumpiness increases, the memory optimization percentage increases. This is due to the map-based approach limiting the redundancy of the shared neighbors. The existing array-based approach creates a new entry for each colocated instance, while the map-based approach only creates a new key for a pattern if the instance of the base pattern is not a pre-existing key. In clumpier data sets, most instances have a large neighborhood, so the map-based approach saves the most memory due to the large number of neighbor relationships and hence the large number of repeated instances. For example, if (A, B) is a colocation pattern with instances $\{A_1, B_1\}$ and $\{A_1, B_2\}$, then that pattern would be stored in the map-based instance table as $\{(A_1) \rightarrow [B_1, B_2]\}$ instead of entries $[A_1, B_1]$ and $[A_1, B_2]$ in the array-based approach, which has redundant entries of A_1.

Time Complexity Analysis: For colocation pattern C_k the time complexity is $O(|I_{k-1}|(kLog(M)+N(Log(M)+k)))$, where k is the cardinality of colocation pattern C_k, I_{k-1} is the average number of entries in the instance table of the previous degree, M is the average length of the star neighborhood for each instance, and N is the average number of neighbors for each key combination.

Table 2. Memory Proportion for Real-World Case Studies

Region	Degree	Map-Based Approach	Array-Based Approach	Memory Optimization
Middle East (1)	2	0.0017 GB	0.0033 GB	48.5%
	3	0.1178 GB	0.3408 GB	65.4%
	4	3.4795 GB	13.0831 GB	73.4%
	Total	3.5990 GB	13.4272 GB	73.2%
North America (2)	2	0.0006 GB	0.0011 GB	48.2%
	3	0.0134 GB	0.0395 GB	65.9%
	Total	0.0140 GB	0.0406 GB	65.5%
South Asia (3)	2	0.0185 GB	0.0362 GB	49.0%
	3	4.0720 GB	11.9411 GB	65.9%
	4	438.4528 GB	1662.6944 GB	73.6%
	Total	442.5433 GB	1674.6717 GB	73.6%

Table 3. Interesting Patterns

Area	Interesting Patterns
Saudi Arabia	(0, 2)
Yemen	(0, 1, 2), (0, 1, 6), (1, 2, 6)
Middle East	(1, 3), (0, 1, 2, 6)
United States	(2, 5), (3, 7), (1, 2, 7)
Mexico	(0, 6), (2, 6), (3, 5)
North America	(1, 6), (0, 1, 2), (0, 2, 3)
Afghanistan	(3, 6), (1, 2, 3), (2, 3, 6)
Pakistan	(1, 4), (1, 5), (0, 2, 3)
South Asia	(0, 1, 2, 3), (0, 1, 2, 6), (0, 2, 3, 6)

7 Conclusion and Future Work

This paper investigates an approach to estimate the optimal spatial neighborhood relationship constraint for colocation mining, as well as a map-based colocation mining algorithm. We evaluated our proposed framework on synthetic data sets and three case studies based on real-world data sets from the Global Terrorism Database. We found that our map-based approach uses approximately 70% less memory than the existing array-based instance table approach and performs noticeably better than the existing array-based approach when the data is clumpy. We also performed time complexity analysis and theoretical analysis on the correctness and completeness of our framework. We present the outcomes of our algorithm applied to diverse data sets across various regions, including the estimated spatial neighborhood relationship constraint of each region and their interesting patterns. Limitations of our work include potential redundancies in the instance table keys as the maximal colocation pattern cardinality increases as well as not accounting for the distance threshold varying from sub-region to sub-region.

In the future, we will explore how to further optimize intermediate data storage, specifically in higher degrees to cut back on redundant keys, and a framework that can account for differing sub-regional spatial neighborhood relationship constraints.

8 Acknowledgements

Our work was partially supported by NSF (DMS-2110826).

References

[1] [n. d.]. statista. http://www.statista.com. Accessed: 2024-02-09.
[2] Kevin Arvai. 2020. kneed. https://kneed.readthedocs.io/en/stable/api.html#kneelocator. Accessed: 2024-02-10.
[3] Berkay Aydin, Dustin Kempton, Vijay Akkineni, Rafal Angryk, and Karthik Ganesan Pillai. 2015. Mining spatiotemporal co-occurrence patterns in solar datasets. *Astronomy and computing* 13 (2015), 136–144.
[4] Jiannan Cai, Qiliang Liu, Min Deng, Jianbo Tang, and Zhanjun He. 2018. Adaptive detection of statistically significant regional spatial co-location patterns. *Computers, Environment and Urban Systems* 68 (2018), 53–63.
[5] Mete Celik, James M Kang, and Shashi Shekhar. 2007. Zonal co-location pattern discovery with dynamic parameters. In *Seventh IEEE International Conference on Data Mining (ICDM 2007)*. IEEE, 433–438.
[6] Min Deng, Jiannan Cai, Qiliang Liu, Zhanjun He, and Jianbo Tang. 2017. Multi-level method for discovery of regional co-location patterns. *International Journal of Geographical Information Science* 31, 9 (2017), 1846–1870.
[7] Min Deng, Qiliang Liu, Tao Cheng, and Yan Shi. 2011. An adaptive spatial clustering algorithm based on Delaunay triangulation. *Computers, Environment and Urban Systems* 35, 4 (2011), 320–332.
[8] Wei Ding, Christoph F Eick, Jing Wang, and Xiaojing Yuan. 2006. A framework for regional association rule mining in spatial datasets. In *Sixth International Conference on Data Mining (ICDM'06)*. IEEE, 851–856.
[9] Jiangli Duan, Wang Lizhen, Xin Hu, and Hongmei Chen. 2018. Mining spatial dynamic co-location patterns. *Filomat* 32, 5 (2018), 1491–1497.
[10] Christoph F Eick, Rachana Parmar, Wei Ding, Tomasz F Stepinski, and Jean-Philippe Nicot. 2008. Finding regional co-location patterns for sets of continuous variables in spatial datasets. In *Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems*. 1–10.

[11] Ahmad Basheer Hassanat, Mohammad Ali Abbadi, Ghada Awad Altarawneh, and Ahmad Ali Alhasanat. 2014. Solving the problem of the K parameter in the KNN classifier using an ensemble learning approach. *arXiv preprint arXiv:1409.0919* (2014).

[12] Yan Huang, Shashi Shekhar, and Hui Xiong. 2004. Discovering colocation patterns from spatial data sets: a general approach. *IEEE Transactions on Knowledge and data engineering* 16, 12 (2004), 1472–1485.

[13] Yan Huang, Hui Xiong, Shashi Shekhar, and Jian Pei. 2003. Mining confident co-location rules without a support threshold. In *Proceedings of the 2003 ACM symposium on Applied computing*. 497–501.

[14] Krzysztof Koperski and Jiawei Han. 1995. Discovery of spatial association rules in geographic information databases. In *International Symposium on Spatial Databases*. Springer, 47–66.

[15] Pradeep Mohan, Shashi Shekhar, James A Shine, James P Rogers, Zhe Jiang, and Nicole Wayant. 2011. A neighborhood graph based approach to regional co-location pattern discovery: A summary of results. In *Proceedings of the 19th ACM SIGSPATIAL international conference on advances in geographic information systems*. 122–132.

[16] Yasuhiko Morimoto. 2001. Mining frequent neighboring class sets in spatial databases. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. 353–358.

[17] Feng Qian, Kevin Chiew, Qinming He, and Hao Huang. 2014. Mining regional co-location patterns with k NNG. *Journal of Intelligent Information Systems* 42 (2014), 485–505.

[18] Feng Qian, Qinming He, Kevin Chiew, and Jiangfeng He. 2012. Spatial co-location pattern discovery without thresholds. *Knowledge and information systems* 33 (2012), 419–445.

[19] Feng Qian, Liang Yin, Qinming He, and Jiangfeng He. 2009. Mining spatio-temporal co-location patterns with weighted sliding window. In *2009 IEEE international conference on intelligent computing and intelligent systems*, Vol. 3. IEEE, 181–185.

[20] Arpan Man Sainju, Danial Aghajarian, Zhe Jiang, and Sushil Prasad. 2018. Parallel grid-based colocation mining algorithms on GPUs for big spatial event data. *IEEE Transactions on Big Data* 6, 1 (2018), 107–118.

[21] Arpan Man Sainju and Zhe Jiang. 2017. Grid-based colocation mining algorithms on gpu for big spatial event data: A summary of results. In *Advances in Spatial and Temporal Databases: 15th International Symposium, SSTD 2017, Arlington, VA, USA, August 21–23, 2017, Proceedings 15*. Springer, 263–280.

[22] Venkatesan Meenakshi Sundaram, Prabhavathy Paneer, et al. 2012. Discovering co-location patterns from spatial domain using a delaunay approach. *Procedia engineering* 38 (2012), 2832–2845.

[23] Venkatesan Meenakshi Sundaram and Arunkumar Thangavelu. 2015. A Delaunay diagram-based min–max CP-tree algorithm for spatial data analysis. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 5, 3 (2015), 142–154.

[24] Lizhen Wang, Yuzhen Bao, Joan Lu, and Jim Yip. 2008. A new join-less approach for co-location pattern mining. In *2008 8th IEEE International Conference on Computer and Information Technology*. 197–202. https://doi.org/10.1109/CIT.2008.4594673

[25] Lizhen Wang, Yuzhen Bao, and Zhongyu Lu. 2009. Efficient discovery of spatial co-location patterns using the iCPI-tree. *The Open Information Systems Journal* 3, 1 (2009).

[26] Jin Soung Yoo and Mark Bow. 2012. Mining spatial colocation patterns: a different framework. *Data Mining and Knowledge Discovery* 24 (2012), 159–194.

[27] Jin Soung Yoo and Shashi Shekhar. 2006. A joinless approach for mining spatial colocation patterns. *IEEE Transactions on Knowledge and Data Engineering* 18, 10 (2006), 1323–1337.

[28] Jin Soung Yoo, Shashi Shekhar, John Smith, and Julius P Kumquat. 2004. A partial join approach for mining co-location patterns. In *Proceedings of the 12th annual ACM international workshop on Geographic information systems*. 241–249.