Seventh IEEE International Conference on Data Mining

Zonal Co-location Pattern Discovery with Dynamic P ers

Mete Celik * James M. Kang Shashi Shekhar

Department of Computer Science, University of Minnesota, MN, USA
{mcelik, jkang, shekhar}@cs.umn.edu

Abstract

Zonal co-location patterns represent subsets of feature-types that are frequently located in a subset of space (i.e., zone). Discovering zonal spatial co-location patterns is an important problem with many applications in areas such as ecology, public health, and homeland defense. However, discovering these patterns with dynamic parameters (i.e., repeated specification of zone and interest measure values according to user preferences) is computationally complex due to the repetitive mining process. Also, the set of candidate patterns is exponential in the number of feature types, and spatial datasets are huge. Previous studies have focused on discovering global spatial co-location pat- with a fixed interest measure threshold. In this paper, opose an indexing structure for co-location patterns ropose algorithms (Zoloc-Miner) to discover zonal co- on patterns efficiently for dynamic parameters. Extensive experimental evaluation shows our proposed approaches are scalable, efficient, and outperform naïve alternatives.

1. Introduction

Spatial data mining and spatial analysis techniques are playing a major role in spatial database systems to discover interesting but implicit patterns in spatial datasets of ever increasing size and complexity. Extracting useful and interesting patterns from massive spatial datasets is important for many application domains, including ecology (e.g., discovering the interactions between species and the interactions between vegetation types), public health (e.g., determining sources of pandemic diseases), and homeland defense (e.g., looking for significant or unusual "events"). Most data mining algorithms assume that patterns are uniformly distributed over the space. This assumption violates the spatial heterogeneity law of Geography: "results of analysis vary from one place to another" [4].

*Corresponding author.

Given a collection of boolean spatial feature-types, their instances over a common spatial framework, a neighbor relation, and a subset of a spatial framework (i.e., zone), a zonal co-location pattern mining algorithm aims to discover correct and complete sets of interesting and non-trivial spatial co-location patterns while minimizing the computation cost. For example, in ecology, there may be several zonal co-location patterns, e.g., symbiotic relationship, predator-prey interactions. The association between crocodiles and birds where a bird will eat pieces of meat between a crocodile's teeth is an example of symbiotic species. The African crocodile and Egyptian Plover in specific regions of Africa form such an association, i.e., a zonal co-location pattern. The interaction between a wolf and its diet of a variety of animals (e.g., elk, caribou, moose, rodent, bison, etc.) is an example of a predator-prey pattern. This predator-prey relationship may differ significantly from one place to another based on the availability of the prey.

However, mining zonal co-location patterns with dynamic parameters (i.e., repeated specification of zone and interest measure values according to user preferences) is challenging for several reasons. First, the repetitive mining process is computationally very expensive due to the dynamic set of parameters. Second, discovering patterns to support dynamic parameters is challenging due to lack of co-location indexing structures. Third, the set of candidate patterns increases exponentially with the number of feature-types. Finally, since spatial datasets are huge, computationally efficient and scalable algorithms are necessary.

Related Work: Previous research on spatial co-location pattern mining has focused on discovering global co-location patterns based on a fixed interest measure. Morimoto [7] used a *support* measure to discover frequently neighboring class sets. Huang et al. developed join-based [5] and Yoo et al. developed partial-join [9] and join-less [10] co-location algorithms using a fixed interest measure (i.e., spatial prevalence measure). Zhang et al. developed fast co-location mining algorithms using multi-way spatial joins [11]. Huang et al. proposed a projection based co-location pattern mining paradigm by extending the FP-tree structure [6] based on an interest measure parameter.

1550-4786/07 $25.00 © 2007 IEEE
DOI 10.1109/ICDM.2007.102

However, these approaches mainly focused on the discovery of global co-location patterns in a spatial dataset, due to the lack of user control on specifiying zones of interest. The results of previous approaches may not represent or capture the characteristics of different zones inside the spatial dataset. Even if co-location patterns of the zone of interest are captured, their interest measure (i.e., spatial prevalence) values will be affected by the instances of the feature-types that are found in the spatial dataset as a but which may not be found in the zone of interuse the spatial prevalence of the patterns may differ erent zones. Recently Ding et. al. defined probregional association rule mining problem and proposed methods discovering regional associations by identifying zones automatically [3]. In this approach, the main focus was identifying interesting subregions, e.g., zones, in spatial datasets for which regional association rules are then generated. Overall, previous approaches do not provide adequate support for efficient pattern mining for changing user parameter specifications. These approaches might need to re-compute the patterns for each set of parameters. In contrast, we define the problem of zonal co-location pattern mining, propose an index structure based on Quad-trees [8] to support dynamic parameters, and develop computationally efficient methods to mine these patterns.

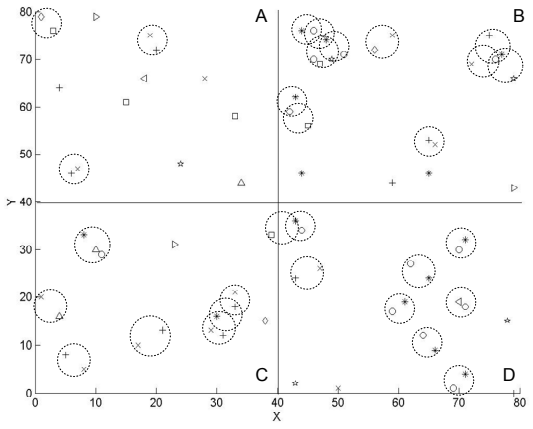


Figure 1. Example dataset

Table 1. Global and Zonal Co-locations

	Global co-locations		Zonal co-locations					
			Zone A	Zone B			Zone C	Zone D
Patterns	$*$,○	+, x	+, x	$*$,○	○,□	○,⋆	+, x	$*$,○
PI	0.786	0.615	0.667	0.667	0.6	0.6	0.8	0.857

To illustrate the difference between global and zonal coon patterns, we will use spatial dataset given in Fig. The dataset gives feature-types and their instances he space. Each feature-type is represented by a dishape. The dataset has four zones (A, B, C, and D). These zones may represent topographic boundaries (e.g., four different districts in a city, etc.) or may represent the regions divided by an indexing approach (e.g., Quad-tree, R-tree, etc.) [8]. Each dashed circle shows the co-located instances that form a clique. Previous approaches will discover global patterns {$*$, ○} and {+, x} for spatial prevalence index threshold 0.5 and neighbor distance 5, because they focus on the whole dataset (Table 1). However, the strength or existence of the global co-locations may vary spatially for each zone. For example, in zone A, pattern {+, x} is frequently co-located with a stronger spatial prevalence measure. In zone B, patterns {$*$, ○}, {○, □}, and {○, ⋆} are frequently co-located. Patterns {○, □}, and {○, ⋆} are not global co-locations, i.e., specific to zone B. In zone C and D, patterns {+, x} and {$*$, ○} are frequently co-located zonal patterns respectively. Also, the prevalence measures of patterns {+, x} and {$*$, ○} are increased (Table 1).

Contributions: First, we define zonal co-location patterns and give a formal definition of the zonal co-location pattern mining problem. Second, we propose a novel and computationally efficient zonal co-location pattern mining algorithm (Zoloc-Miner). Third, we propose an index structure for storing co-location patterns to handle dynamic parameters (i.e., changing user parameter specifications). Finally, we experimentally evaluate the proposed algorithms.

The rest of the paper is organized as follows. Section 2 presents basic concepts and defines the problem of mining zonal co-location patterns. Section 3 discusses the challenges of zonal co-location pattern discovery and presents our proposed algorithm. The experimental evaluation is given in Section 4. We conclude and name directions for future work in Section 5.

2. Basic Concepts and Problem Statement

In this section, first we explain the modeling of co-location patterns in space and then, we explain how we model zonal co-location patterns with dynamic parameters.

2.1 Basic Concepts

Spatial co-location mining algorithms are used to discover subsets of feature-types that are frequently located together in space for a given set of feature-types, their instances, and a neighbor relation R [5]. If instances form a clique for a given neighbor relation, they are co-located. A **spatial prevalence measure**, e.g., participation index (PI), is used to determine the strength of the co-location pattern, that is, whether the index is greater than or equal to a given threshold [5]. Such a co-location is called **prevalent**. PI is defined as the minimum of the **participation ratios** (the fraction of the number of instances of feature-types forming co-location instances to the total number of instances). For more information, we refer readers to [5].

2.2 Modeling Zonal Co-location Patterns

This section presents several relevant definitions to our proposed approaches.

Definition 2.1 *Given a spatial framework, a **zone** is a subset of the spatial framework.*

Definition 2.2 *Given a zone, a set of spatial feature-types, their instances, and a neighbor relation R, a **zonal co-location pattern** is a subset of feature-types whose instances are frequently neighbors in the given zone.*

Definition 2.3 *Given a zone and a neighbor relation R, a **buffer** is the surrounding area of the zone R distance.*

Definition 2.4 *Given neighboring co-location instances and a zone with its buffer, a **cross-neighbor** is a co-location instance where at least one of the co-location features is in the buffer zone.*

For example, in Figure 1, the co-location instance $\{o, *\}$ is a cross-neighbor since feature o is in zone C and feature $*$ is in the buffer of zone C.

Lemma 2.1 *Given two zones S and T, if zone P is the union of zones S and T, i.e. $P = S \cup T$, the co-location patterns of zone P are the union of the co-locations of zone S, zone T, the buffer of zone S, and buffer of zone T that are intersecting with zone P.*

Proof: The co-location patterns and their instances of zones S and T will be generated from the spatial points where all the points in zones S and T respectively. The co-location patterns of the buffer of zone S will include co-location instances where at least one of the spatial points are in the buffer. The co-location patterns of the buffer of zone T will include co-location instances where at least one of the spatial points of instances are in the buffer. Buffers will capture the cross-neighbor co-located patterns. □

For example in Figure 1, suppose zone P is the bottom half of the given dataset, that is, the union of zones C and D. In addition to the co-location pattern instances of zones C and D, zone P will include buffer instance $\{o, *\}$, which is a cross-neighbor.

3. Mining Zonal Co-location Patterns

The basic idea of zonal co-location pattern mining is to index space while storing relevant co-location patterns. It is possible to use any of the co-location mining approaches developed in the literature [5, 9, 10, 11]. However, known spatial indexing structures are not directly applicable for co-location pattern mining. In this paper, we propose an index structure to mine zonal co-location patterns with dynamic parameters efficiently. Some of the major challenges of designing an index structure for zonal co-location patterns are:

(i) *Discovering cross-neighboring co-locations:* It is possible that co-located patterns are split and stored in different nodes of the index structure. This makes it hard to discover the co-location patterns of parent nodes, and may lead to separate patterns and is addressed in our proposed index structure (Section 3.2) by adding a buffer (Definition 2.3) for each indexed zone. Index not only stores co-location instances of zone of interest but also cross-neighboring instances (Definition 2.4).

(ii) *Computational complexity:* As the number of points increases, the computational complexity of the co-location algorithm increases due to the generation of all possible candidates and is addressed by running the algorithm in the leaf nodes and their buffers of the tree (Section 3.2).

(iii) *Overlapping user-defined mining zones:* If a user-defined zone fits one of the nodes of the tree structure, the algorithm will output patterns of this zone, since patterns are stored in the tree structure. If a user-defined zone overlaps more than one node of the tree, the challenge is how to discover co-located patterns across multiple or partial nodes and is addressed in Section 3.3 based on Lemma 2.1.

A discussion of a naïve approach to mine zonal co-location patterns is given in Section 3.1. The naïve approach initially indexes the spatial framework using the classical Quad-tree and performs the general co-location method on the zone of interest. Due to the repeated specificatons of zones of interest and interest measure values according to user preferences, an excessive amount of computation time is needed to re-calculate the co-locations for each zone of interest and interest measure values. In contrast, we propose to discover co-locations of each indexed space within a new index structure called a clQuad-tree (Section 3.2). Then, an algorithm can be used to discover the co-locations of each leaf of the tree (Section 3.3).

3.1 Naïve Approach

The naïve approach to disco o-location patterns has two phases. The first phase aims to index the space using a spatial indexing structure. Given a zone, the index structure will retrieve the area to be mined. The second phase aims to retrieve the zone of interest from the index structure. Then a co-location mining algorithm [5, 9, 10, 11] can be used to discover the prevalent patterns in that zone using a prevalence threshold. For dynamic parameters, i.e., repeated specification of zone and PI threshold values, the second phase of the naïve approach will be repeated several times. This leads to unnecessary computational costs.

Algorithm 1 Bulk Load of clQuad-tree

```
1: Function BULKCLQT(Neighbor R, set points)
   {Phase I: Create initial clQuad-tree}
2: for each object o ∈ points do
3:     insert o into clQuad-Tree
4: end for
   {Phase II: Insert points into buffer regions}
5: for each object o ∈ points do
6:     if a quad q does not contain o then
7:         Increase q size by R
8:         if q contains object o then
9:             Insert o into quad q's Buffer Region
10:        end if
11:    end if
12: end for
   {Phase III: Generate co-locations for each quad}
13: for each quad q ∈ clQuad-tree do
14:     Generate candidate patterns using quad q object-types
15:     Discover pattern instances using R from q and Buffer Regions
16:     Store patterns as object instances and pattern labels for q
17: end for
```

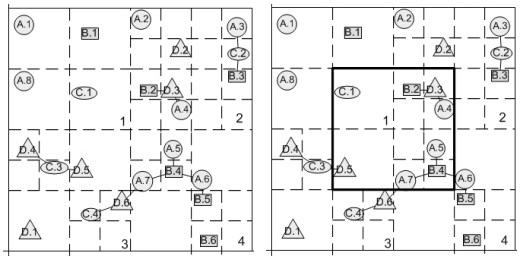
3.2 clQuad-Tree Bulk Load Algorithm

This section presents the bulk load algorithm for the proposed index structure clQuad-tree (Algorithm 1) that maintains an index of spatial co-located patterns. In general, the clQuad-tree is built in three phases: (1) the initial quads are created within the clQuad-Tree, (2) points belonging to the quads' buffer regions are assigned, and (3) the co-locations are generated and stored for each quad in the clQuad-Tree.

Phase 1-Create Initial clQuad-tree : In this phase, all possible quads are created for the next two subsequent steps. Each point p in the set $Points$ is inserted into the clQuad-Tree (Lines 2-4 of Algorithm 1). The insertion process is the same as in the classical Quad-Tree [8].

Phase 2-Create Buffer Regions : In this phase, the aim is to ensure that potential co-located patterns, i.e., cross-neighbors, are not split because of the quad boundaries. Thus, a buffer is added to each quad for a given neighbor distance R by investigating each point object o in the dataset (Line 5 of Algorithm 1). If a quad in the clQuad-Tree does not contain this object, then the quad region is increased by a factor of the distance R only once (Lines 6-7 of Algorithm 1). After the quad region has been expanded, if this object o is now contained in region, then this object is added to this quad (Lines 8-9 of Algorithm 1). Otherwise object is not added to the quad. This process will continue until all points have been added to their respective quad buffer regions (Lines 5-12 of Algorithm 1).

Phase 3-Generate Co-locations : In this phase, the co-located patterns and their instances of each zone are generated (Lines 13-17 of Algorithm 1). First, candidate patterns are generated (Line 14 of Algorithm 1) for each quad and buffer. Second, the instances of the candidates are discovered for a given distance R (Line 15 of Algorithm 1). Finally, the patterns and their instances are stored as a simple table (Line 16 of Algorithm 1). Once the patterns and instances are generated for the leaf nodes and buffers of the tree, the patterns and instances of the parent nodes are found by taking the union of the patterns and instances of leaf nodes and their buffers (Lemma 2.1).



(a) Quad-tree representation (b) Overlapping zone

Figure 2. Example dataset for execution trace

Execution Trace: The execution trace of the proposed algorithm is discussed using the dataset in Figure 2. The first phase of clQuad-tree indexes the dataset to capture not only the dataset, but the co-locations as well. This dataset contains four object-types A, B, C, and D and their instances over the space. A, B, C, and D has 8, 6, 4, and 6 instances respectively. Figure 2(a) gives the quad tree representation of the dataset. The lines between features shows the co-located instances that are satisfying distance R. The second phase identifies the spatial points that fall into the buffers of each leaf node of the clQuad-tree. The normal and buffer points of zones 1, 2, 3, and 4 are given in Table 2. In the third phase, patterns of the indexed zones are discovered. The pattern of a parent node is the union of its child nodes and their buffers (Lemma 2.1). To find the patterns of the root node of the clQuad-tree, the patterns of zones 1, 2, 3, and 4 and their buffers should be found (Table 3). Zone 1 and its buffer has no pattern. Table 4 gives the patterns of the root node which is the unions of the patterns and instances of the child nodes of the root that are given in Table 3.

Table 2. Normal and buffer points of zones

Zone 1		Zone 2		Zone 3		Zone 4	
Normal points	Buffer points	Normal points	Buffer points	Normal points	Buffer points	Normal points	Buffer points
A.1	A.2	A.2	A.5	C.3	A.7	A.5	A.4
B.1	B.2	A.3		C.4		A.6	D.6
C.1	D.4	A.4		D.1		A.7	
A.8		B.2		D.4		B.4	
		B.3		D.5		B.5	
		C.2		D.6		B.6	
		D.2					
		D.3					

Table 3. Zonal co-location patterns

Co-location	Zone 2				Zone 3		Zone 4	
	Normal				Normal	Buffer	Normal	Buffer
Patterns	A, C	A,D	B, C	B, D	C, D	A, D	A, B	A, D
Instances	<A.3,C.2>	<A.4,D.3>	<B.3,C.2>	<B.2,D.3>	<C.3,D.4> <C.3,D.5> <C.4,D.6>	<A.7,D.6>	<A.5,B.4> <A.6,B.4> <A.7,B.4>	<A.7,D.6>

Table 4. Co-location patterns of the root

Co-location	Root of Quad Tree					
Patterns	A, B	A, C	A, D	B, C	B, D	C, D
Instances	<A.5,B.4> <A.6,B.4> <A.7,B.4>	<A.3,C.2>	<A.4,D.3> <A.7,D.6>	<B.3,C.2>	<B.2,D.3>	<C.3,D.4> <C.3,D.5> <C.4,D.6>

Algorithm 2 Zonal Pattern Mining using clQuad-tree

```
1: Function SEARCHCLQT(MiningZone r, clThreshold θ)
   {Phase I: Zonal Co-Location Pattern Retrieval}
2: cand ← ∅
3: for each quad q intersecting Z do
4:     for each pattern p ∈ q do
5:         if object instances o_i for p intersect r then
6:             cand ← cand ∪ o_i
7:         end if
8:     end for
9: end for
   {Phase II: Find Prevalent Co-location Patterns}
10: ans ← ∅
11: for each pattern p ∈ cand do
12:     Calculate participation index PI for p
13:     if PR > θ then
14:         ans ← ans ∪ p
15:     end if
16: end for
17: return ans
```

3.3 Zoloc-Miner Algorithm

This section presents the zonal co-location pattern mining algorithm (Algorithm 2) utilizing the clQuad-tree. The Zoloc-Miner is performed in two phases: (1) a set of potential candidates are retrieved for a given zone, and (2) patterns are discovered that satisfy the prevalence threshold θ.

Phase 1-Zonal Co-location Pattern Retrieval : In phase 1, co-location instances for a given zone are retrieved from the tree. Each quad q that intersects with the zone Z is inspected (Line 3 of Algorithm 2). Each quad consists of a set of patterns and their instances. If the instances of a pattern intersect the zone, then the pattern is added to the $cand$ set (Line 6 of Algorithm 2). This process continues for all quads intersecting with the zone (Lines 3-9 of Algorithm 2).

Phase 2-Find Prevalent Patterns : In phase 2, prevalent patterns that satisfy prevalence threshold θ are determined. Initially, for each pattern p in the candidate $cand$ set, the PI is calculated (Lines 11-12 of Algorithm 2). If the PI of a pattern is no less than the PI threshold θ, then the pattern is added to the ans set. Finally, the algorithm outputs the set of prevalent zonal co-location patterns ans.

Execution Trace : There may be two zones of interest to discover the zonal co-location patterns. If the user-defined zone fits one of the leaf nodes of the tree, Zoloc-Miner will retrieve relevant pattern instances and spatial points of this leaves. Then, the PIs of patterns will be calculated and the ones that do not satisfy the PI threshold θ will be pruned. For example, assume that zone 2 (Figure 2(a)) is the mined area with the PI threshold 0.5. In zone 2, there are 4 candidate patterns and in that zone: A has 3 instances (i.e., A.2, A.3, and A.4), B has 2 instances (i.e., B.2, B.3), C has 1 instance (i.e., C.2), and D has 2 instances (i.e., D.2, D.3). Patterns {A,C} and {A,D} will be pruned since their PIs are 1/3 and 1/3 respectively. Patterns {B,C} and {B,D} are discovered since they satisfy the PI threshold of 0.5. If the given zone overlaps more than one node of the clQuad-tree, the points and pattern instances that are intersecting with the zone of interest will be retrieved. Then, non-prevalent patterns will be pruned. For example, if the zone with the bolded outline in Figure 2(b) is the mined area having the PI threshold of 0.5, then there are 3 candidates and zone A has 3 instances (i.e., A.4, A.5, and A.7), B has 2 instances (i.e. B.2, B.4), C has 1 instance (i.e., C.1), and D has 2 instances (i.e., D.3, D.5). Patterns {A,B}, {A,D} and {B,D} are discovered since they satisfy the PI threshold of 0.5.

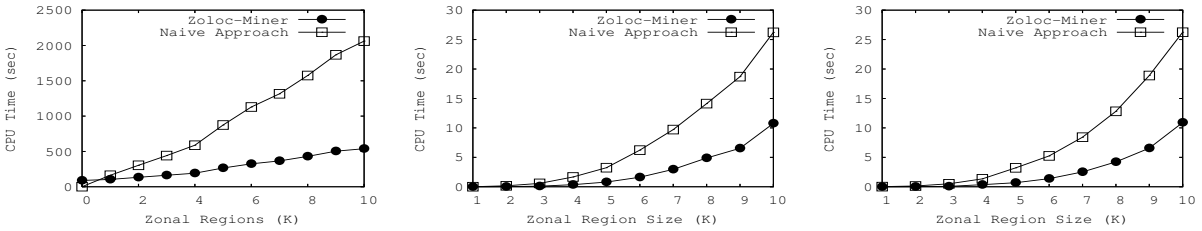
4 Experimental Evaluation

We evaluated the behavior of the naïve approach and Zoloc-Miner using synthetic datasets to answer the following questions: (i) What is the effect of the number of zones? (ii) What is the effect of the size of a zone? (iii)What is the effect of the amount of overlap in a zone?

Synthetic datasets were generated based on the spatial data generator proposed in [5]. To generate the datasets, frame size, neighboring distance, number of feature-types, average number of instance of features were set at 10^4, 10, 100, and 10 respectively. Average co-location size and number of maximal patterns were set at 10 and 4 respectively. The ratio of noise features over number of features was set at 0.25. The number of the noise instances was set at 500. In the experiments, the PI threshold and distance were set at 0.3 and 10 respectively. Experiments were conducted on an Intel Pentium IV CPU 2.0GHz with 512MB RAM.

1) Effect of Number of Zones: Figure 3(a) gives execution times of both algorithms for varying number of zones. We ran the experiments up to 1000 zones. The size of each zone was assigned randomly. As can be seen, the execution time of both algorithms increases, as the number of zones increased. The naïve approach gives better performance at the first set of zonal regions, this is because the clQ tree requires more time to pre-compute and to store zonal co-location patterns. Later, Zoloc-Miner runs faster since patterns are already stored in the clQuad-tree.

2) Effect of Size of Zone: Figure 3(b) gives the effect of the size of the zones and the execution time for both algorithms. The size of the zones represents the diagonal distance between the minimum x and y-coordinates and the maximum x and y-coordinates. The minimum x and y-coordinates were fixed at (0,0) and the maximum ordinates was incremented by 1K. At zone size < gorithms have similar performance. This is due hat there were similar numbers of patterns in t iner and number of objects in the naïve approach. At zone size >3K, Zoloc-Miner outperform the naïve approach. This is be-



(a) (b) (c)

Figure 3. (a)Number of Zonal Regions Cost, (b) Varying Zonal Region, (c) Varying Zonal Overlap

cause the number of candidates exponentially increase as the number of objects increase from the larger zone.

3) Effect of Zonal Overlap: Figure 3(c) gives the effect of varying the amount of zonal overlap and execution time for both algorithms. The zonal overlap refers to maximizing the ar ersection with the quad regions in the index met zone size is the diagonal distance between the and y-coordinates and the maximum x and y-c However, the zone at 1K is started at the minimum x and y-coordinates and maximum x and y-coordinates of (45K,45K) and (55K,55K) respectively and incremented by 1K by expanding the zone. At zone size $<$3K, both approaches have similar performance due to the limited number of zonal co-location patterns and feature-types within the zone. At zone sizes $>$ 3K, the Zoloc-Miner outperform the naïve approach, which generates unnecassary candidates as the region increases.

5. Conclusions and Future Work

We defined the zonal co-location patterns and mining problem. We developed a novel computationally efficient zonal co-location pattern mining algorithm (Zoloc-Miner) for mining these patterns. We developed an indexing structure (clQuad-tree) to store co-locations and their instances and to handle dynamic parameters, i.e., changing user parameter specifications. The proposed Zoloc-Miner is compared with the Naïve approach, which indexes the space using a classical Quad-tree and runs the co-location mining algorithm for each user-defined zones by retreiving intersecting quads. We also evaluated the proposed algorithms experimentally. As future work, we plan to explore new structures to optimize the storage of the patterns. We also plan to extend our proposed algorithm for mining spatio-temporal patterns [1, 2].

6 Acknowledgements

This work was partially supported by the US Army Corps of Engineers under contract number W9132V-06-C-0011, the NSF grant ISS-0431141 and the NSF IGERT grant. The authors would like to thank Kim Koffolt for her comments.

References

[1] H. Cao, N. Mamoulis, and D. W. Cheung. Discovery of collocation episodes in spatiotemporal data. In *ICDM*, pages 823–827, Hong Kong, China, 2006.

[2] M. Celik, S. Shekhar, J. P. Rogers, J. A. Shine, and J. S. Yoo. Mixed-drove spatio-temporal co-occurrence pattern mining: A summary of results. In *ICDM*, pages 119–1287, Hong Kong, China, 2006.

[3] W. Ding, C. F. Eick, J. Wang, and X. Yuan. A framework for regional association rule mining in spatial datasets. In *ICDM*, pages 851–856, Hong Kong, China, 2006.

[4] M. F. Goodchild. The fundamental laws of giscience. In *Keynote address, Annual Assembly, University Consortium for GIScience*, Monterey, CA, June 2003.

[5] Y. Huang, S. Shekhar, and H. Xiong. Discovering co-location patterns from spatial datasets: A general approach. *TKDE*, 16(12):1472–1485, 2004.

[6] Y. Huang, L. Zhang, and P. Yu. Can we apply projection based frequent pattern mining pradigm to spatial co-location mining? In *PAKDD*, pages 719–725, Vietnam, 2005.

[7] Y. Morimoto. Mining frequent neighboring class sets in spatial databases. In *ACM SIGKDD*, pages 353 – 358, 2001.

[8] H. Samet. *Foundations of Multidimensional and Metric Data Structures*. Morgan-Kaufmann, 2006.

[9] J. S. Yoo and S. Shekhar. A partial join approach for mining co-location patterns. In *ACM-GIS*, pages 241–249, USA, 2005.

[10] J. S. Yoo and S. Shekhar. A joinless approach for mining spatial colocation patterns. *TKDE*, 18(10):1323–1337, 2006.

[11] X. Zhang, N. Mamoulis, D. W. L. Cheung, and Y. Shou. Fast mining of spatial collocations. In *ACM SIGKDD*, pages 384–393, Seatle, WA, 2004.