World Wide Web (2019) 22:1971–1997
https://doi.org/10.1007/s11280-018-0646-2

Mining maximal sub-prevalent co-location patterns

Lizhen Wang[1] · Xuguang Bao[1] · Lihua Zhou[1] · Hongmei Chen[1]

Received: 19 April 2018 / Revised: 23 September 2018 / Accepted: 31 October 2018 /
Published online: 2 February 2019

Abstract
Spatial prevalent co-location pattern mining is to discover interesting and potentially useful patterns from spatial data, and it plays an important role in identifying spatially correlated features in many domains, such as Earth science and Public transportation. Existing approaches in this field only take into account the clique instances where feature instances form a clique. However, they may neglect some important spatial correlations among features in practice. In this paper, we introduce *star participation instances* to measure the prevalence of co-location patterns such that spatially correlated instances which cannot form cliques will also be properly considered. Then we propose a new concept called *sub-prevalent co-location patterns (SPCP)* based on the star participation instances. Furthermore, two efficient algorithms – the prefix-tree-based algorithm (PTBA) and the partition-based algorithm (PBA) – are proposed to mine all the *maximal* sub-prevalent co-location patterns (MSPCP) in a spatial data set. PTBA uses a typical candidate generate-and-test way starting from candidates with the longest pattern-size, while PBA adopts a step-by-step manner starting from 3-size core patterns. We demonstrate the significance of our proposed new concepts as well as the efficiency of our algorithms through extensive experiments.

Keywords Spatial data mining · Spatial co-location pattern · Sub-prevalent co-location pattern (SPCP) · Star participation ratio (SPR) · Star participation index (SPI)

This article belongs to the Topical Collection: *Special Issue on Web Information Systems Engineering 2017*
Guest Editors: Lu Chen and Yunjun Gao

✉ Hongmei Chen
hmchen@ynu.edu.cn

Lizhen Wang
lzhwang@ynu.edu.cn

Xuguang Bao
bbaaooxx@163.com

Lihua Zhou
lhzhou@ynu.edu.cn

[1] School of Information Science and Engineering, Yunnan University, Kunming, China

1 Introduction

As the rapid development of LBS (Location-Based Services) and the easier access of processing techniques for spatial information, spatial data containing enormous number of spatial information with various useful attributes has become available [1, 10]. This kind of data includes VLSI chip layout data, preprocessed remote sensing or medical imaging data, and geographic search logs comprising with associated locations, etc.

A spatial co-location pattern represents subsets of spatial features (or spatial objects, attributes) whose instances are frequently located close together, and the spatial co-location pattern mining aims to uncover the interesting and frequent co-located spatial features in various domains. For example, a co-location pattern may show that the region where mosquitoes are abundant and poultry are kept usually has West Nile virus around; or can find that the percentage of sub-humid evergreen broadleaved forests growing with orchid plants can reach 80% [6, 25].

The traditional model for discovery of prevalent co-location patterns was firstly proposed by Shekhar and Huang [6, 16] where the prevalence measure of a co-location pattern is defined based on *clique instances* under spatial neighbor relationships. In other words, a prevalence measurement called PI (participation index) is defined as the measurement for the prevalence of a co-location pattern, the PI value of a co-location c is the minimum participation ratio $Pr(f_i, c)$ among all features f_i in c; $Pr(f_i, c)$ is defined as the fraction of the instances of f_i participating in row-instances (set of instances having clique relationship) of c. Figure 1 shows an example spatial data set containing four spatial features (A, B, C, and D), if the corresponding distance between two spatial instances is no more than a distance threshold d, a line connects them in Figure 1. An instance written as $f.i$ denotes the i-th instance of feature f. In Figure 1, the co-location pattern {A, B, C} only has one row-instance {A.1, B.1, C.1}, thus, if the given prevalence threshold is more than 1/2, the co-location pattern {A, B, C} is not considered prevalent. But by looking into the distribution of this pattern {A, B, C}, three instances of A neighbor to instances of the other features, and B has two instances neighboring to instances of the other two features, and C has two instances similarly. That is to say, at least 40% (min{3/4, 2/5, 2/3}) of instances of each spatial feature

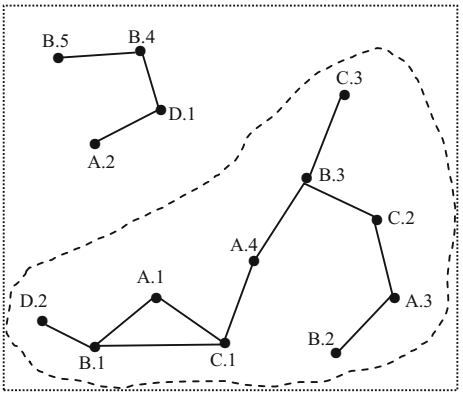


Figure 1 An example of the spatial data sets

neighbor to the other features' instances in {A, B, C}. This reveals that instances of the three features A, B and C form spatially correlated relationships, which might be neglected if only considering the clique instances.

Based on above, patterns like {A, B, C} in Figure 1 may contain meaningful distribution of instances which can help decision makers gain further insights. For example, given a vegetation distributional data set, if there exists a pattern similar to {A, B, C} in Figure 1, then the other vegetations present in the neighborhood of growing any vegetation in {A, B, C}, i.e., vegetations A, B and C represent symbiotic vegetation. This kind of patterns shows significance in practice. For instance, they can be well used in analysis of vegetation distribution, site selection for investigating vegetations, as well as vegetation protection, etc.

On the other hand, we consider Figure 1 as a data set in real daily life, thus, the features A, B and C represent "hospital", "resident area" and "bus station", respectively. By the observation of the distribution of instances of the three features, it can be seen that B.3 neighbors to A.4, C.2 and C.3, while A.4, C.2 and C.3 are not neighbors to any one of the others, but A.4 and C.1 are neighbors. In fact, the three features within the dotted line in Figure 1 have satisfied a co-located correlation because for each instance except the endpoints (B.2, C.3 and D.2) in the dotted line, there are instances of the other two features connected.

In the traditional model for co-location pattern discovering, to measure the prevalence of a co-location pattern c, each row-instance of c needs to be traversed. Note that the concept of row-instance is a clique-based concept. However, the statement "the presence of B and C in the neighborhood of an instance of feature A" only demonstrates B and C must neighbor to A, the neighbor relationship of B and C is not necessary. Thus, in this paper, a new concept called *sub-prevalent co-location patterns* was defined to follow the statement above by replacing row-instance with star participation instances.

In traditional co-location pattern mining, each row-instance of c must form a clique in neighborhood relationship, while for a sub-prevalent co-location, clique relationship is not necessary. Thus, a prevalent co-location pattern is a sub-prevalent pattern while not vice versa. In other words, the set of prevalent co-locations discovered from a spatial data set are a subset of sub-prevalent co-location set discovered from the same data set. However, the large amount of discovered sub-prevalent co-locations may increase the burden of the user and cause the user easily confused. By means the *downward inclusion property* satisfied by sub-prevalent co-locations, we consider study the discovery of *maximal* sub-prevalent co-location patterns (MSPCP) to effectively reduce the volume of sub-prevalent co-locations.

This paper's main contributions can be summarized as follows:

First, instead of clique relationship for each instance of a co-location, a star neighborhood relationship was proposed to get an instance (called star participation instance) of a co-location. Using star participation instances, a new concept called sub-prevalent co-location patterns (SPCPs) was presented. This concept can better describe the frequently co-located features in practice.

Second, two novel algorithms, named prefix-tree-based algorithm (PTBA) and partition-based algorithm (PBA), are proposed to discover MSPCPs in this paper. Furthermore, an intersection-based method is proposed to compute the prevalence of patterns with star participation instances.

Third, strength and weakness of the proposed algorithms are analyzed in depth, as well as their computational complexities.
Finally, sufficient experiments are deliberately designed to evaluate our approaches. The experimental results show that our algorithms are scalable and can discover the potential star-correlations of the spatial features which cannot be discovered by traditional models.

The remainder of this paper is organized as follows. Section 2 gives the definitions on the discovery of Sub-Prevalent Co-location Patterns (SPCPs), and proves the *anti-monotonic property* of SPCPs. Two novel algorithms for discovering MSPCPs are designed in Section 3 and 4, respectively. The experimental results are shown in Section 5 and Section 6 gives the related work. Finally, the conclusion is given in Section 7.

2 Basic concepts and properties

In this section, we first review the basic concept of prevalent co-location pattern (PCP) mining, and then present the concept of the MSPCP mining and discuss its anti-monotonic property.

2.1 Prevalent co-location patterns (PCPs)

Given a set of spatial features (a kind of things, e.g. KFC) $F = \{f_1, f_2, \ldots, f_n\}$ where each feature f_i has some instances (an occurrence of a feature, e.g. KFC branch No. 2), suppose S is the set containing all instances of features in F. We use a *spatial neighbor relationship* R to describe the relationships of two instances, i.e. If two instances i_1 and i_2 are neighbors to each other, the relationship can be denoted as $R(i_1, i_2)$. In common works, Euclidean distance are commonly used to define the relationships, that is, for two instances i_1 and i_2, if the distance between them is no more than a given distance threshold d ($distance(i_1, i_2) \leq d$), they are neighbors ($R(i_1, i_2)$). We use Figure 1 as an example spatial data set, where two instances are connected with a line if they are neighbors. This data set contains four features A, B, C and D and their instances, A.1 means the first instance of A observed. In figure, A has four instances (A.1, A.2, A.3 and A.4). A *Spatial co-location pattern* c is a subset of F, $c \subseteq F$, whose instances form cliques frequently under R. A *row-instance* I of a co-location c is a set of instances containing instances of all the features in c and forming a clique relationship, and any subset of I cannot contain all the features in c. The set containing all row-instances in c is called *table-instance* of c. The *size* of c is the number of features in c.

For example, in Figure 1, {B.1, D.2} is a row instance of a 2-size co-location pattern {B, D} but {B.5, D.1} is not, and the table-instance $T(\{B, D\})$ is {{B.1, D.2}, {B.4, D.1}}.

In co-location pattern mining, *participation index* (*PI*) is widely used to measure the prevalence of co-location patterns [1–3, 6, 8, 18, 25–32]. Before defining PI, we first give the definition of participation ratio (PR).

Definition 1 Participation Ratio (PR).

Given a co-location pattern c, the *participation ratio* of a feature f_i in c, denoted as $PR(c, f_i)$, is the fraction of instances of feature f_i that participate in $T(c)$. That is,

$$PR(c, f_i) = \frac{\text{Number of distinct instances of } f_i \text{ in } T(c)}{\text{Total number of instances of } f_i} \quad (1)$$

Definition 2 Participation Index (PI).

The *participation index* of a co-location pattern c is defined as

$$PI(c) = \min_{fi \in c}\{PR(c, f_i)\} \quad (2)$$

Definition 3 Prevalent Co-location Pattern (PCP).

Given a user-specified prevalence threshold *min_prev*, a co-location pattern c is a *prevalent co-location pattern* (PCP) if $PI(c) \geq min\_prev$.

For example, in Figure 1, $T(\{B, D\}) = \{\{B.1, D.2\}, \{B.4, D.1\}\}$. The participation ratio of feature B in {B, D}, $PR(\{B, D\}, B)$, is 2/5 since two out of five instances of B participate in $T(\{B, D\})$. In the same way, $PR(\{B, D\}, D)$ is 2/2. Thus, the participation index of {B, D}, $PI(\{B, D\})$, is $min\{PR(\{B, D\}, B), PR(\{B, D\}, D) = 2/5 = 0.4$. If the prevalence threshold *min_prev* is set as 0.3, the co-location {B, D} is prevalent.

The PI and PR measures satisfy the *anti-monotonicity property* (*downward closure property*), i.e., $PI(c) \geq PI(c')$ for any $c \subset c'$, and $PR(c, f) \geq PR(c', f)$ for any $c \subset c'$ *and* $f \in c$ [6].

Based on the anti-monotonicity of co-location prevalence, we can give the concept of maximal prevalent co-location patterns, which infers the original collection of PCPs.

Definition 4 Maximal Prevalent Co-location Pattern (MPCP).

A prevalent co-location pattern (PCP) c is *maximal* if there is no PCP c' such that $c \subset c'$.

2.2 Sub-Prevalent Co-location Patterns (SPCPs)

By replacing clique instances with star neighborhood instances, we introduce the concept of Sub-Prevalent Co-location Patterns (SPCPs). The related definitions are as follows.

Definition 5 Star Neighborhoods Instance (SNsI).

$SNsI(i_j) = \{i_k \mid distance(i_j, i_k) \leq d,$ where d is a given distance threshold$\}$ is the star neighborhoods instance of i_j. In other words, $SNsI(i_j)$ is a set comprising instances in its neighborhood and itself as the center instance.

In Figure 1, $SNsI(A.4) = \{A.4, B.3, C.1\}$, and $SNsI(C.2) = \{A.3, B.3, C.2\}$. Based on Definition 5, the following concepts are defined successively: star participation instances (SPIns), star participation index (SPI) as well as star participation ratio (SPR) to measure the prevalence of instances from different features in a co-location pattern using star neighbors.

Definition 6 Star Participation Instance (SPIns).

SPIns$(c, f_i) = \{i_j \mid i_j$ is an instance of feature f_i and *SNsI*(i_j) includes instances of all features in $c\}$ is the star participation instance of feature f_i in c. In other words, *SPIns*(c, f_i) is the set of instances of f_i whose instances in its star neighborhoods contain instances of all features in c.

Definition 7 Star Participation Ratio (SPR).

$SPR(c, f_i) = |SPIns(c, f_i)|/|S_{f_i}|$ represents the star participation ratio of feature f_i in a co-location c, where S_{f_i} is the set containing all instances of f_i. In other words, *SPR*(c, f_i) is the ratio of instances of f_i that occur in the star participation instance of f_i in c to the whole instances of f_i.

Definition 8 Star Participation Index (SPI).

$SPI(c) = \min_{f_i \in c}\{\mathrm{SPR}(c, f_i)\}$ is the star participation index of a co-location c, which is the minimum star participation ratio *SPR*(c, f_i) among all features f_i in c.

Definition 9 Sub-Prevalent Co-location Pattern (SPCP).

A co-location c is a sub-prevalent co-location pattern (SPCP), if and only if its star participation index is no less than a given sub-prevalence threshold *min_sprev*, i.e., $SPI(c) \geq min\_sprev$.

For example, in Figure 1, for a co-location pattern $c = \{A, B, C\}$, the star participation instance *SPIns*$(c, A) = \{A.1, A.3, A.4\}$, *SPIns*$(c, B) = \{B.1, B.3\}$, and *SPIns*$(c, C) = \{C.1, C.2\}$. Thus, the star participation ratio *SPR*$(c, A) = 3/4$, *SPR*$(c, B) = 2/5$, *SPR*$(c, C) = 2/3$, and $SPI(c) = min\{3/4, 2/5, 2/3\} = 2/5$. If *min_sprev* is set as 0.4, c is a SPCP.

Lemma 1 (Monotonicity of SPR and SPI). Let c and c' be two co-location patterns ($c' \subseteq c$). Then, for each feature $f \in c'$, $SPR(c', f) \geq SPR(c, f)$. Furthermore, $SPI(c') \geq SPI(c)$.

Proof For the first claim in the lemma, it only needs to prove that for a spatial feature $f \in c'$, $|SPIns(c', f)| \geq |SPIns(c, f)|$.

Since $c' \subseteq c$, every star participation instance of feature f in c includes instances of all features in c'. Thus, the inequality holds.

The second claim follows from the fact that $SPI(c') = \min_{f \in c'}\{SPR(c', f)\} \geq \min_{f \in c}\{SPR(c, f)\} = SPI(c)$.

Lemma 1 declares that both SPR and SPI satisfy the downward inclusion property.

Intuitively, based on the introduction of the concept of SPCP, longer co-location patterns can be discovered. But the massive volume and mutually inclusive of the discovered results may cause the user confused and costly on the calculation of redundant results. Thus, to solve this disadvantage, a concept of maximal sub-prevalent co-location patterns is defined to concisely express the massive results.

Definition 10 Maximal Sub-Prevalent Co-location Pattern (MSPCP).

Given a SPCP $c = \{f_l, \ldots, f_v\}$ in a set of spatial features $F = \{f_1, f_2, \ldots, f_n\}$, $l, v \in \{1, 2, \ldots, n\}$, if any of c's super-patterns is not sub-prevalent, c is called a Maximal Sub-Prevalent Co-location Pattern (MSPCP).

Similar with the maximal co-location patterns [28] defined based on traditional prevalent co-locations, the defined MSPCPs are the minimum representation that can not only express the prevalence of all SPCPs but also reduce the volume of the whole SPCPs effectively.

In the following two sections, two novel algorithms focus on the discovery of complete MSPCPs are proposed, respectively.

3 A prefix-tree-based algorithm

A prefix-tree-based algorithm (PTBA) is proposed in this section. In PTBA, in order to efficiently perform candidate pruning, all MSPCP candidates generated from 2-size SPCPs are organized into a *prefix tree*. Meanwhile, an *intersection-based* method is presented to compute the star participation index (SPI) of candidates. Finally, the performance of PTBA is analyzed, and then a pruning lemma is given.

3.1 Basic idea

To demonstrate the basic idea of PTBA, first, star neighborhoods instance is equivalent to each other, i.e., if $o_j \in SNsI(o_i)$, $o_i \in SNsI(o_j)$. Thus, we firstly collect pairs of instances neighbors, and then select 2-size SPCPs by comparison with a given sub-prevalence threshold *min_sprev*.

With Lemma 1, from the set of 2-size SPCPs, MSPCP candidates can be generated using feature sets forming cliques. For example, Given a set of 2-size SPCPs: $SPCP_2$ = {{A, B}, {A, D}, {A, E}, {B, C}, {B, D}, {B, E}, {C, D}, {D, E}}, the feature set {A, B, D, E} is a 4-size MSPCP candidate because all its 2-size subsets are sub-prevalent, while {C, D, E} is not a candidate because {C, E} is not included in the set of 2-size SPCPs.

A lexicographic order based method is used to generate all MSPCP candidates. For reducing the candidate search space during the candidate generation process, a *prefix tree* is constructed to organize all generated candidates where each branch expresses a candidate and new branches would share common prefixes. For example, the prefix tree constructed using a set of candidates {{ABCD, ABC, ABD, ACD, AC, AD}, {BCD, BC, BD}, {CDE, CE}, {DE}} is shown in Figure 2, the candidate {A, B, C, D} is the first candidate to be added to the candidate search space tree. The following candidate {A, B, C} cannot form a new branch because {A, B, C} is a prefix of {A, B, C, D},

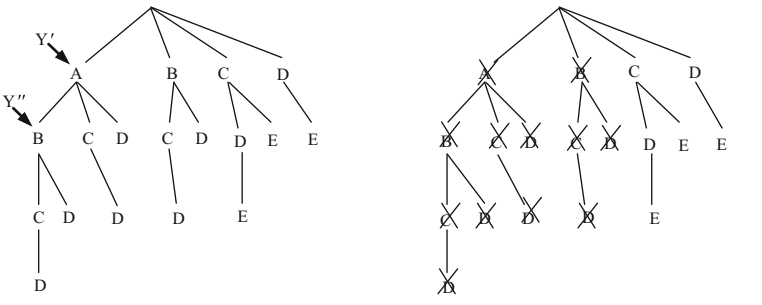


Figure 2 Candidate search space tree and pruning

whilst {A, B, D} can form a new branch because only {A, B} of {A, B, D} can be represented by {A, B, C} or {A, B, C, D}.

To identify each node of the prefix tree, two attributes *head* and *tail* are assigned to each node, the *head* attribute demonstrates its ancestor nodes and the *tail* attribute indicates the set of its descendant nodes. For example, consider nodes Y' and Y'' in Figure 2 whose *head* attribute values are {A} and {A, B}, respectively, and the *tail* attribute value are the set {B, C, D} and {C, D}, respectively. Thus, the Head Union Tail (HUT) of Y' is {A, B, C, D}.

A maximal sub-prevalent co-location pattern is a sub-prevalent co-location pattern that any of its supersets is not sub-prevalent while all its subsets are sub-prevalent. Thus, it is reasonable to generate MSPCP candidates from long size to short size, and the longest MSPCP candidate is firstly to be checked whether it is a MSPCP according to Definition 9 and 10. Once a candidate is checked as a MSPCP, the algorithm then starts a pruning process. Each node in the prefix tree is traversed using a breadth-first way to check whether its HUT is a subset of a current maximal set. If its HUT is a subset of any current MSPCPs, the sub-tree whose root is the node can be pruned. For example, in Figure 2a, suppose that {A, B, C, D} is a MSPCP. The breadth-first way of checking and pruning for candidates in the search space tree is performed as follows. First, the HUT of node A is {A, B, C, D} which is a subset of {A, B, C, D}, thus, the sub-tree whose root is A is pruned. The next node in same level is node B with HUT {B, C, D} also a subset of {A, B, C, D}, thus, this sub-tree can be pruned. The pruning process continues with the rest of the nodes in the first level. Figure 2b shows the candidate search space tree after the checking and pruning operations when the maximal set only contains {A, B, C, D}.

The processes mentioned above perform continually until there are no more candidates to process.

From the method discussed above, the core problem is to compute the sub-prevalence measure of a candidate c, i.e. SPI(c) (star participation index of c). Using the 2-size co-location instances (neighborhoods instances' pairs), according to Lemma 2, the SPI of any k-size pattern ($k > 2$) can be computed.

Lemma 2 Given the 2-size co-location instances of a spatial data set, the SPI of a k-size co-location pattern $c = \{f_1, f_2, \ldots, f_k\}$ can be calculated as follows:

$$SPI(c) = min\left\{ \left| \bigcap_{j=2,\ldots k} SPIns\left(\left\{f_1, f_j\right\}, f_1\right)\right| / |S_{f_1}|, \ldots, \left| \bigcap_{j=1,\ldots k-1} SPIns\left(\left\{f_k, f_j\right\}, f_k\right)\right| / |S_{f_k}| \right\} \quad (3)$$

where $SPIns(c, f_i)$ is the set of star participation instances of f_i in c.

Proof For $c = \{f_1, f_2, \ldots, f_k\}$, according to Definition 6, $SPIns(c, f_i) = \bigcap_{j=1,\ldots k, j\neq 1} SPIns\left(\left\{f_i, f_j\right\}, f_i\right)$. And by Definitions 7 and 8, the formula for calculating $SPI(c)$ is obvious.

For example, in Figure 1, If c = {A, B, C}, SPI(c) = min{|{A.1, A.3, A.4} ∩ {A.1, A.3, A.4}| / 4, |{B.1, B.2, B.3} ∩ {B.1, B.3}| / 5, |{C.1, C.2} ∩ {C.1, C.2, C.3}| / 3} = min{3/4, 2/5, 2/3} = 0.4.

3.2 Algorithm

The pseudo-code of PTBA to mine all MSPCPs is shown in Algorithm 1.

Algorithm 1 Prefix-tree-based Mining Algorithm

Input

$F=\{f_1,f_2,\ldots,f_n\}$: a set of spatial feature types;
S: a spatial data set;
d: a spatial neighbor distance threshold;
min_sprev: a minimum sub-prevalence threshold;

Output

MSPCP: A set of all MSPCPs;

Variables

SI_2: a set of 2-size co-location instances;
l: a co-location size of interest;
C: a set of all candidate sets;
C_l: a set of l-size candidates;
spi: Star participation index;
l_{max}: the longest size of candidate c;
$MSPCP_l$: a set of size l MSPCPs;
MSPCP: a set of all MSPCPs;

Steps

Preprocess and candidate generation

1. SI_2=gen_2-size_co-loc_ins (S, F, d);
2. $SPCP_2$=sel_2-size_sub-prev_co-loc (*min_sprev*, SI_2);
3. *MSPCP=null*;
4. C=gen_maxi_candi ($SPCP_2$); //they are organized into a prefix tree

MSPCP mining

5. l_{max} =longest_size(C);
6. $l = l_{max}$;
7. **while** ($l > 2$ and $C_l \neq \phi$) **do**
8. C_l = get_l_candi (C, l);
9. **for** each candidate c in C_l **do**
10. *spi* = calculate_spi(c, SI_2); //using Lemma 2
11. **if** *spi* ≥ *min_sprev*
12. Insert(c, $MSPCP_l$);
13. **end for**
14. $MSPCP = MSPCP \cup MSPCP_l$;
15. Subset_Pruning($MSPCP_l$, C);
16. $l = l - 1$;
17. **end while**

Preprocessing and candidate generation (Step 1–4) Given an input spatial data set and a neighbor distance threshold d, we firstly find all neighboring instance pairs (all 2-size co-location instances) using a geometric method such as plane sweep, or a spatial query method of using quaternary trees or R-Trees. We then compute their star participation index (SPI) and determine whether a 2-size co-location is sub-prevalent, by comparing SPI to a user-defined sub-prevalence threshold *min_sprev*, and include it as the result set of $MSPCP_2$. Based on $MSPCP_2$, all MSPCP candidates are generated using the lexicographic order method.

Select l-size (from l_{max} to 2) co-location candidates (Step 5–8) First, the longest size of candidates is set to l_{max}. The MSPCP mining is processed from size $l = l_{max}$. Select l-size candidates C_l from the candidate pool.

Calculate l-size sub-prevalent co-locations (Step 9–13) For each candidate c in C_l, based on the related 2-size co-location instances SI_2, SPI(c) is calculated by Lemma 2. If SPI(c) ≥ *min_sprev*, insert c into the l-size MSPCP set $MSPCP_l$.

Update result set and prune the subsets (Step 14–15) Update the maximal sub-prevalent result set *MSPCP*, and all subsets of the maximal set $MSPCP_l$ are pruned.

Return the final result set (Step 16–17) The procedure from Step 7 to Step 17 is repeated continually until $l = 3$ or the candidate set is empty. The final result then can be returned.

3.3 Analysis and pruning

The main cost of performing the prefix-tree-based algorithm (PTBA) occurs with procedures *gen_2-size_co-loc_ins*, *gen_max_candi*, and the loop of Step 7. Suppose the total number of spatial instances is m, the number of features is n, and A^m denotes the average number of instances in all features. Then the cost of procedure *gen_2-size_co-loc_ins* is at most $O(m^2 \log_2 m)$, and procedure *gen_max_candi* is at most $O(n^2)$ if we suppose that computing an order string needs only a unit time by the lexicographic order method.

For the loop of Step 7, the procedures *calculate_spi* and *Subset_Pruning* are the dominant costs. If we calculate the SPI of candidates using Lemma 2, the procedure *calculate_spi* would cost at most $O(\sum_{k=3}^{l_{max}} |C_k| \cdot k^2 \cdot A^m)$, where $|C_k|$ is the number of k-size candidates. The computational complexity of procedure *Subset_Pruning* is related to the number of candidates, and the effects of pruning. The smaller is the size of MSPCPs, the fewer is the number to be pruned, and so higher is the cost of *Subset_Pruning*.

For a data set containing n features, the number of candidates in some extreme situation may reach as high as 2^n, and with a short average size of candidates, the effect of procedure *Subset_Pruning* may not be ideal. Thus, the following pruning lemma is proposed to efficiently prune sub-trees or nodes in a candidate search space tree when a candidate is not a MSPCP.

Lemma 3 (Depth-first pruning). If a candidate $c = c' \cup \{X\} \cup \{Y\}$ is not maximal sub-prevalent,

(1) The node *"Y"* can be pruned;
(2) If $SPI(c' \cup \{X\} \cup \{Y\}) = SPI(c' \cup \{X\})$, the sub-tree whose root is node *"X"* can be pruned;
(3) If $SPI(c') = SPI(c' \cup \{X\})$, the node *"Y"* which is a brother of the node *"X"* can be pruned.

Proof Case (1) is obviously satisfied because the candidate c is not a MSPCP; For case (2) $SPI(c' \cup \{X\} \cup \{Y\}) = SPI(c' \cup \{X\})$, because $SPI(c' \cup \{X\}) = SPI(c' \cup \{X\} \cup \{Y\}) < min\_sprev$, according to case (1), the sub-tree whose root is "X" can be pruned; For case (3) $SPI(c') = SPI(c' \cup \{X\})$, because $SPI(c' \cup \{Y\}) = SPI(c' \cup \{X\} \cup \{Y\}) < min\_sprev$, according to case (1), the node "Y" in $c' \cup \{Y\}$ can also be pruned.

For example, Figure 3a and b respectively show the pruning results of cases (2) and (3) in Lemma 3 after finding the non-MSPCP {A, C, D, E}.

In this section, PTBA algorithms has been proposed to discover all MSPCPs, although it is feasible, in big and dense data sets, large number of candidates may be generated from the candidate search space tree, meanwhile, the process for checking sub-prevalence of a pattern is cost, these problems limit the performance of PTBA algorithm. Thus, to solve above problems efficiently, a novel partitioning technique will be proposed in the next section.

4 A partition-based algorithm

4.1 Method

In this section, a novel method called partition-based algorithm (PBA) is proposed where a divide-and-conquer strategy is adopted. This strategy can be stated as follows. First, the 2-size SPCPs are divided into a set of strings by the relation $=_{head}$ in lexicographic order, and then a *core pattern method* is used for each string to do the mining process separately. We first define the following related concepts.

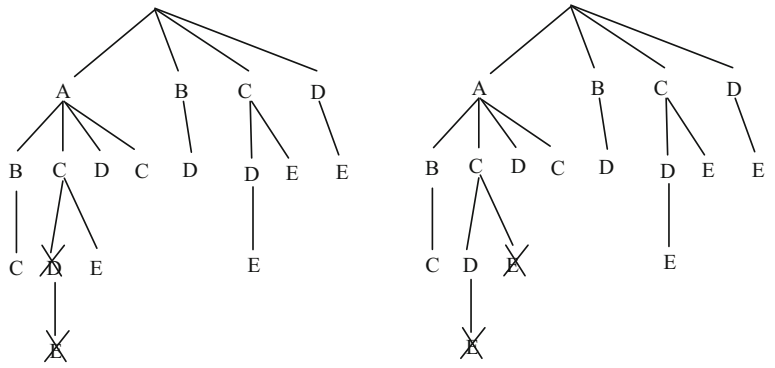


Figure 3 Pruning results after finding {A, C, D, E} is not maximally sub-prevalent

Definition 11 Partition pattern (PP) and core pattern (CP).

A *partition pattern* (PP) of a co-location c is a 2-size pattern contained in c, if its star participation index (SPI) is the biggest of all 2-size co-locations of c, and two SPCPs divided by the partition pattern (PP), are called *core patterns* (CPs) of c.

For example, given a co-location $c = \{B, D, E, G\}$, if the PP of c is $\{D, E\}$, CPs of c are $\{B, D, G\}$ and $\{B, E, G\}$. Because PP is a 2-size co-location of c having the biggest SPI value, the CPs divided by PP are more likely to be key patterns determining the sub-prevalence of c.

The process of the *core pattern method to identify SPCPs:* Given a l-size co-location $c = \{f_1, f_2, \ldots, f_l\}$ and its PP $\{f_{l-1}, f_l\}$, c can be divided into two (l-1)-size CPs $\{f_1, \ldots, f_{l-2}, f_{l-1}\}$ and $\{f_1, \ldots, f_{l-2}, f_l\}$ by $\{f_{l-1}, f_l\}$. If two CPs $\{f_1, \ldots, f_{l-2}, f_{l-1}\}$ and $\{f_1, \ldots, f_{l-2}, f_l\}$ are sub-prevalent, and the they satisfy the following two additional conditions, c is a SPCP.

The additional condition (1):

$$|SPIns(\{f_1,...,f_{l-2},f_{l-1}\},f_{l-1})\cap SPIns(\{f_{l-1},f_l\},f_{l-1})|/|S_{f_{l-1}}|\geq \min\_sprev$$
$$\text{and } |SPIns(\{f_1,...,f_{l-2},f_l\},f_l)\cap SPIns(\{f_{l-1},f_l\},f_l)|/|S_{f_l}|\Big\}\geq min\_sprev$$

The additional condition (2):

$$\min\Big\{|SPIns(\{f_1,...,f_{l-2},f_{l-1}\},f_1)\cap SPIns(\{f_1,...,f_{l-2},f_l\},f_1)|/|S_{f_1}|,...,$$
$$|SPIns(\{f_1,...,f_{l-2},f_{l-1}\},f_{l-2})\cap SPIns(\{f_1,...,f_{l-2},f_l\},f_{l-2})|/|S_{f_{l-2}}|\Big\}\geq min\_sprev$$

In fact,

$$SPI(c)=\min\Big\{|SPIns(\{f_1,...,f_{l-2},f_{l-1}\},f_1)\cap SPIns(\{f_1,...,f_{l-2},f_l\},f_1)|/|S_{f_1}|,...,$$
$$|SPIns(\{f_1,...,f_{l-2},f_{l-1}\},f_{l-2})\cap SPIns(\{f_1,...,f_{l-2},f_l\},f_{l-2})|/|S_{f_{l-2}}|,$$
$$|SPIns(\{f_1,...,f_{l-2},f_{l-1}\},f_{l-1})\cap SPIns(\{f_{l-1},f_l\},f_{l-1})|/|S_{f_{l-1}}|,$$
$$|SPIns(\{f_1,...,f_{l-2},f_l\},f_l)\cap SPIns(\{f_{l-1},f_l\},f_l)|/|S_{f_l}|\Big\}$$

The idea of this method is shown in Figure 4 visually.

For identifying two CPs $\{f_1,\ldots,f_{l-2},f_{l-1}\}$ and $\{f_1,\ldots,f_{l-2},f_l\}$, they can be treated recursively by the core pattern method employing a bottom-up iteration strategy starting from identifying core patterns using a level-wised way, i.e., starts from 3-size core patterns, then 4-size,…, until l-size patterns. The following example gives an illustration on how the PBA algorithm works.

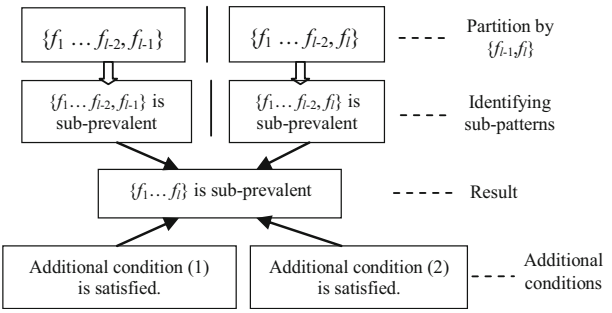


Figure 4 The idea of core pattern method

Suppose there is the set of 2-size SPCPs: $SPCP_2 = \{\{B, C\}, \{B, D\}, \{B, E\}, \{B, G\}, \{B, H\}, \{C, D\}, \{C, E\}, \{C, H\}, \{D, E\}, \{D, G\}, \{D, H\}, \{E, F\}, \{E, G\}, \{E, H\}, \{F, G\}\}$ in a spatial data set with feature set {B, C, D, E, F, G, H}.

First, $SPCP_2$ is partitioned into a set of strings (denoted as L_1) in lexicographical order under the relation $=_{head}$. Thus, $L_1 = \{\delta_{head}(B) = BCDEGH, \delta_{head}(C) = CDEH, \delta_{head}(D) = DEGH, \delta_{head}(E) = EFGH, \delta_{head}(F) = FG, \delta_{head}(G) = G, \delta_{head}(H) = H\}$. This partition is called **Partition_1**.

Then, using 2-size non-SPCPs, ordered strings in L_1 are divided further. In our example, $\overline{SPCP_2}\{BF, CF, CG, DF, FH\}$. Thus, $\delta_{head}(B) = BCDEGH$ is further divided into two strings BCDEH and BDEGH by CG included both in non-$SPCP_2$ and $\delta_{head}(B)$. The division process will be continually performed on above sub-strings successively until no 2-size or more non-SPCPs in it. Thus, L_1 can be replaced with $L = \{BCDEH, BDEGH, CDEH, DEGH, EFG, EGH, FG\}$. The non-SPCP-based partition is called **Partition_2**.

Next, strings in L are to be treated successively using the core pattern method to obtain the whole MSPCPs. In our example, we first take into account the ordered string "BCDEH". Suppose {D, E} is the PP of $c = \{B, C, D, E, H\}$, then the CPs of c are {B, C, D, H} and {B, C, E, H}. Continually using the partition process, we may finally obtain 3-size CPs {B, C, D} and {B, C, H} for {B, C, D, H}, and {B, C, E} and {B, C, H} for {B, C, E, H}. The partition obtaining CPs based on PP is called **Partition_3**. The real line part in Figure 5 is the result of partitioning {B, C, D, E, H} by Partition_3.

Given a 3-size core pattern, it can be computed directly using Lemma 2, and the SPI value of k-size ($k > 3$) can be computed by k-1-size core patterns. In our example:

Given two 3-size core patterns {B, C, D} and {B, C, H}, if they are sub-prevalent and satisfy additional conditions (1) and (2), the 4-size co-location {B, C, D, H} combined by {B, C, D} and {B, C, H} is sub-prevalent. Similarly, we can identify {B, C, E, H}, and then identify {B, C, D, E, H} using {B, C, D, H} and {B, C, E, H}.

If a 5-size pattern {B, C, D, E, H} is not sub-prevalent, its 4-size patterns with the same head B (two CPs excluded) {B, C, D, E} and {B, D, E, H} (dotted line part in Figure 5) are required to be identified. Note that a lower-size pattern may be used for identifying its higher-size supersets for multiple times. For example, in Figure 5, the pattern {B, C, H} is used to identify its supersets {B, C, D, H} and {B, C, E, H}, and the pattern {B, C, D} is used to identify its supersets {B, C, D, E} and {B, C, D, H}. Thus, if the results of the lower-size patterns are stored once they are identified, it can save much time for identifying its higher-size supersets.

If the process of checking the sub-prevalence of ordered string "BCDEH" is finished, the ordered string "BDEGH" in $\delta_{head}(B)$ is then to be checked. Next, we obtain the set $MSPCP_B$ from MSPCPs with the head "B", and then prune all subsets of $MSPCP_B$ in L. In the following, the ordered strings with a head of "C" in L are handled.

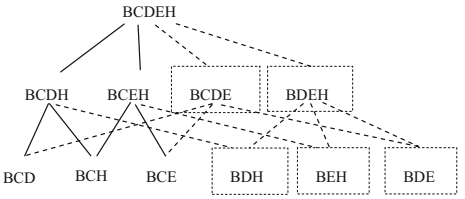


Figure 5 {B, C, D, E, H} and its sub-sets with head of "B"

Algorithm 2 summarizes this mining process.

Algorithm 2 Partition-Based Mining Algorithm

Input and Output is the same as Algorithm 1

Variables

SI_2: a set of 2-size co-location instances; $SPCP_2$: a set of 2-size sub-prevalent co-locations; L: a set of ordered strings of $SPCP_2$ under Partition_1and Partition_2; L_i: a set of ordered strings with the head "i" in L; $MSPCP_i$: a set of MSPCPs with the head of "i"; B: a mark array to indicate a pattern has been identified or not; $MSPCP$: a set of all MSPCPs;

Steps

//**Preprocessing**

1. SI_2=gen_2-size_co-loc_ins (S, F, d);
2. $SPCP_2$=sel_2-size_sub-prev_co-loc (min_sprev, SI_2);
3. $MSPCP$=*null*;
4. L=Partitioning_1_2($SPCP_2$); //by Partition_1 and Partition_2

//**Main program**

5. **for** i**=1 to** m //m features corresponding to digits 1-m
6. L_i =get_head_i(L); //L_i is the set with the head of "i" in L
7. $MSPCP_i$=*null*;
8. mark array B is set initial value 0;
9. **while** ($L_i \neq \phi$) **do**
10. get a pattern c from L_i;
11. HS_c =*null*; //HS_c is the set of all MSPCPs of c
12. CPD(c, $|c|$);
13. $HS_c \leftarrow$ Deal with the results of c;
14. $MSPCP_i = MSPCP_i$ *merge* HS_c;
15. **endwhile**
16. $MSPCP = MSPCP \cup MSPCP_i$;
17. Subset_Pruning($MSPCP_i$, L);
18. **endfor**

Preprocessing (Step 1–4) First, compute the instances set SI_2 of 2-size co-locations, and select all 2-size $SPCP_2$. Then, based on Partition_1 and Partition_2, we obtain a set L of ordered strings.

Get the ordered strings with the head of "i" (Step 5–8) First, the set of ordered strings with the head of "i" in L is put to L_i. The set $MSPCP_i$ of all MSPCPs with the head of "i" is set to "*null*". The mark array B is set initial value 0 (B prevents repeat computation), and the value of B is 1 if the corresponding pattern is sub-prevalent, otherwise −1. "i" runs from the first feature to the last feature.

Deal with ordered strings in L_i by the core pattern method (Step 9–15) For each ordered string c in L_i, compute the set HS_c of all MSPCPs in c by the recursive procedure CPD(c, $|c|$). The design of the recursive procedure is based on the core pattern method. For Step 13, according to the computational results of CPD(c, $|c|$), we can obtain MSPCPs of c. For example, if B(Hash(c)) = 1 then $c \rightarrow HS_c$. *Either* c is a unique MSPCP in c, or else we have

to check the two core patterns c' and c'' of c. Next, we merge the computational result HS_c into the result set $MSPCP_i$ with the head of "i". In the merging operation of Step 14, we need to delete patterns which are the sub-sets of other patterns. Then the next ordered string in L_i is dealt with until there are no ordered strings left.

Update result set and prune the subsets (Step 16–17) Update maximal sub-prevalent result set $MSPCP$, and all subsets of $MSPCP_i$ in L are pruned.

Return the final result set (Step 18) The set of ordered strings with the head of the next feature is dealt with until there are no ordered strings left in L. Finally, return the final result set $MSPCP$ of MSPCPs.

The recursive procedure CPD(c, $|c|$) in Algorithm 2 is shown in below.

```
//Recursion procedure
  Procedure CPD(c, k)
    1. if B(Hash(c))<>0 then Return;      //c has been identified
    2.  if k=3 then
    3.     directly calculate SPI(c) by Lemma 2;
    4.     if SPI(c)≥min_sprev then B(Hash(c))=1;
    5.     else  B(Hash(c))=-1;
    6.  else CPD(c', k-1);              //c' and c'' are two CPs of c
    7.       CPD(c'', k-1);
    8.       if B(Hash(c'))=1 and B(Hash(c''))=1 then
    9.            calculate SPI(c) by core pattern method ;
    10.           if SPI(c)≥min_sprev then B(Hash(c))=1
    11.           else B(Hash(c))=-1;
    12.           for each rest (k-1)-size c''' with the head of "i" in c
    13.               CPD(c''', k-1);
    14.               HS_c ← Deal with the results of c'''
    15.           endfor
    16.      else B(Hash(c))=-1;
    17.           for each rest (k-1)-size c''' with the head of "i" in c
    18.             CPD(c''', k-1);
    19.             HS_c ←Deal with the results of c'''
    20.           endfor
    21.      endif
    22. endif
```

In CPD, if c has been identified (i.e., c is a common pattern) then return; Steps 2–5 exit the recursive procedure; Step 6 and 7 recursively call CPD with the two core patterns c' and c'' of c; Steps 8–15 are for dealing with the case that two core patterns c' and c'' of c are sub-prevalent, while Steps 16–20 deal with the case that c' and c'' are not all sub-prevalent, whereas Steps 12–14 and Steps 17–19 are for dealing with patterns other than the two core patterns, marked as dotted lines in Figure 5.

Analysis of computational complexity The main cost of Algorithm 2 comes from performing Partition_1, Partition_2 and the recursive procedure CPD. The cost of Partition_1 + Partition_2 is about $O(n^2)$ because we scan all 2-size SPCPs in Partition_1, and

because all 2-size non-SPCPs are scanned in Partition_2, the total number of 2-size patterns is $n(n\text{-}1)/2$, where n is the number of spatial features.

For the cost of recursive procedure CPD, if we suppose the worst cost of computing a k-size pattern c is $T(k)$, then $T(k)$ satisfies: $T(k) = \begin{cases} 3 & k = 3 \\ (k-1)T(k-1) & k > 3 \end{cases}$.

A 3-size pattern needs 3 times the intersection operations, and when $k > 3$, in the worst case, all (k-1)-size sub-patterns are recursively called in CPD, and for a k-size pattern c, there are k-1 sub-patterns which are (k-1)-size and with the same head of c.

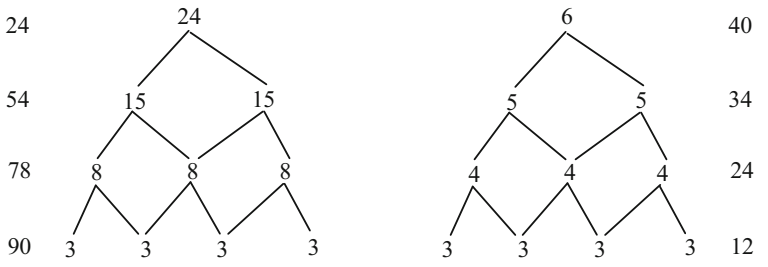
Thus $T(k)$ is about $\prod_{i=3}^{k-1} 3 \cdot i$. This is the worst case and includes large amounts of repeated computation. Repeated computation has been avoided in the core pattern method.

4.2 Comparison with the Prefix-tree-based Method

We have mentioned the main disadvantages of the PTBA algorithm in Section 3: huge volume of candidate search space tree in big and dense data sets and expensive cost for the computation of SPI of candidates by Lemma 2. The PBA algorithm presented in this section aims to solve these two problems.

PBA uses two partitions Partition_1 and Partition_2 to divide all candidates into equivalent classes, in each equivalent class, patterns having the same head are successively to be identified, which can solve the problem of the candidate search space tree being too large.

PTBA algorithm starts the identification of patterns with sizes from long to short, while the PBA method is on the contrary which performs from 3-size to higher sizes. Considering the expected cost of the intersection operations, the core pattern method adopted in PBA is better than the PTBA algorithm. For example, given a 6-size pattern c = {A, B, C, D, E, F}, it needs 24 intersection operations to directly compute SPI(c) by Lemma 2 because 4 intersection operations are needed for the computation of SPIns(c, f) where f is a feature of c. If c is not sub-prevalent, 15 intersection operations are needed to compute a 5-size subset of c, and 4-size subset is 8. A 3-size pattern costs 3 intersection operations by using the core pattern method. To compute a 4-size pattern based on two 3-size core patterns needs 4 intersection operations. Similarly, the cost of a



(a) The computational cost of prefix-tree-based method

(b) The computational cost of core pattern method

Figure 6 The comparison of a 6-size pattern's computation cost

5-size is 5 and 6-size's is 6. As shown in Figure 6a's left and b's right, we respectively list the computational cost attributed to the relevant size. Thus, the expected cost of the PTBA for a 6-size pattern c is $(90 + 78 + 54 + 24)/4 = 61.5$, while by PBA it is $(40 + 43 + 24 + 12)/4 = 27.5$. In fact, we have the following Lemma 4.

Lemma 4 The expected cost ratio of the core pattern method versus the PTBA in identifying a size k ($k > 2$) co-location pattern is about $\frac{2}{k-1}$.

Proof In the following proof, we only consider the computational cost of the two methods where their computational volume is similar to Figure 6, where a 6-size pattern contains two 5-size sub-patterns, three 4-size sub-patterns, and four 3-size sub-patterns.

For the core pattern method, the total cost of a k-size pattern is $\left(\sum_{i=3}^{k} i\cdot(k-i+1)^2\right)$. For example, a 6-size pattern c, it is $(3\cdot4^2 + 4\cdot3^2 + 5\cdot2^2 + 6\cdot1^2) = 110$. So, the expected cost of identifying c is $\left(\sum_{i=3}^{k} i\cdot(k-i+1)^2\right)/(k-2)$.

For the prefix-tree-based method, the expected cost of a k-size pattern is $\left(\sum_{i=3}^{k} i\cdot(i-2)^2\cdot(k-i+1)\right)/(k-2)$. For example, a 6-size pattern c, it is $(3\cdot1^2\cdot4 + 4\cdot2^2\cdot 3 + 5\cdot3^2\cdot2 + 6\cdot4^2\cdot1)/4 = 246/4 = 61.5$.

Consequently, the expected cost ratio τ of two methods is: $\tau = \frac{\sum_{i=3}^{k} i\cdot(k-i+1)^2}{\sum_{i=3}^{k} i\cdot(i-2)^2\cdot(k-i+1)}$. We note that τ is approximately equal to the ratio of their mid-values. That is $\tau \approx \frac{\left(k-\frac{k+3}{2}+1\right)}{\left(\frac{k+3}{2}-2\right)^2} = \frac{2}{k-1}$.

The explanation of formula $\left(\sum_{i=3}^{k} i\cdot(k-i+1)^2\right)$ In the core pattern method, for a k-size pattern c, we need to compute 3 patterns of (k-2)-size and, at least, 4 patterns of (k-3)-size. In general, i patterns of (k-i + 1)-size need to be computed, until we get a k-size pattern c. Furthermore, the computational cost of an i-size pattern is i, and the computation of an i-size pattern is based on (i-1)-size core patterns. So, the cost i of an i-size pattern is repeatedly counted (k-i + 1) times.

The explanation of formula $\left(\sum_{i=3}^{k} i\cdot(i-2)^2\cdot(k-i+1)\right)$ In the prefix-tree-based method, for a k-size pattern c, we also need to compute i patterns ($i = 3,\ldots k$) of (k-i + 1)-size where the computational cost of a i-size pattern is i·(i-2) in the prefix-tree-based method. When we compute a i-size pattern, we must have already computed all higher patterns, so the cost i·(i-2) of a i-size pattern is repeatedly counted (i-2) times.

If a pattern is considered not sub-prevalent, the core pattern method could find it sooner using Partition_3 based on a partition pattern which SPI is the biggest in all the 2-size co-locations with respect to it. In addition, the middle computational results could either be MSPCPs or be used in the computation of other corresponding patterns. For example, in Figure 5, if the computation comes to 5-size {B, C, D, E, H} (i.e., its two 4-size core patterns are sub-prevalent), and that it is not sub-prevalent has been determined, then we can declare that the two core patterns {B, C, D, H} and {B, C, E, H} are maximally sub-prevalent up to now. At the same time, the computational results of 3-size

patterns {B, C, D} and {B, C, E} could also be used to identify the remaining 4-size patterns marked by dotted lines in Figure 5.

5 Experiments

In this section, real data sets and extent synthetic data sets are used to evaluate the performance of PTBA and PBA. In order to better analyze the experimental results, the join-less algorithm [5] has been improved to mine all MPCPs (called M-join-less algorithm), compared with other classical algorithms (e.g. join-based) for co-location pattern mining, M-join-less algorithm handle more data within the same run-time.

All algorithms are memory-based and implemented using C# with Intel i3–3240 @3.4 GHz CPU and 4GB of memory. In all the following experiments, the results of all algorithms presented in this section have been manually verified to be expected results.

5.1 Synthetic data generation

Synthetic data sets were generated using a spatial data generator similar to [6, 32]. Synthetic data sets allow greater control over studying the effects of interesting parameters.

Table 1 describes the parameters used for our data generation. First, we generated F features, and then generated H 2-size core patterns, where the feature types of each 2-size core pattern were randomly chosen from F features. Next, average of P 2-size co-location instances per 2-size core pattern was generated. The total number of instances was N. For locating 2-size instances, we randomly chose S 2-size core patterns as a 2-size cluster, and then chose a point randomly in the spatial frame ($D \times D$), and located a cluster instance within an area ($d \times d$) whose center is the chosen point, and where cluster instances are united by 2-size co-location instances in a cluster.

To generate our specialized data sets, first, the spatial frame size $D \times D$ controls overall data density. For a fixed total number of instances N, the smaller D was, the denser the data was. Second, the data density in neighborhood areas was controlled by a parameter *clumpy*. When a

Table 1 Experimental Parameters and Their Values in Each Experiment

Parameter	Definition	Experimental tables or figures						
		T.2/F.7	F.8	F.9	F.10(a)	F.10(b)	F.10(c)	F.10(d)
H	The number of 2-size core co-locations	60	60	60	60	60	60	60
S	The size of a cluster formed by 2-size core co-locations	6	6	6	6	6	6	6
P	The average number of 2-size instances in 2-size core co-locations	200	200	200	N/100	200	200	200
N	The number of instances	20K,15K	20K	20K	*	20K	20K	20K
F	The number of features	20,15	20	20	20	*	20	20
D	Spatial frame size($D \times D$)	*	1K	2K	2K	2K	5K	1K
d	Neighborhood distance threshold	15	15	15	15	15	*	15
min_sprev	Sub-prevalence threshold	0.3	0.2	0.3	0.3	0.3	0.3	*
clumpy	The number of cluster instances generated in a same neighborhood area	1	1	*	1	1	1	1
overlap	The ratio of points overlapped in different cluster instances over all points	0	*	0	0	0	0	0

*: Variable values; K=1000

point was randomly chosen for locating a 2-size cluster instances, a *clumpy* number of cluster instances were generated in the $d \times d$ area. The default *clumpy* value is 1. Finally, the number of instances overlapped in different cluster instances was controlled by the parameter *overlap*, and $N \times overlap$ instances were randomly selected. The parameter values for the synthetic data set used in each experiment are described in Table 1.

5.2 Comparison of Computational Complexity Factors

In this section, three synthetic data sets were used to test the costs of computational complexity factors with different density: A sparse data set having 20 features and the maximal size of its generated MSPCPs is 6; a dense data set also having 20 features but the maximal size of its generated MSPCPs is as high as 9; a dense* (shown in Table 2) data set where 13-size MSPCPs can be generated from 15 features to better observe the advantage of PTBA compared with PBA. Table 2 shows the total execution time and the percentage of each progress for different methods in different data sets. Overall, it can be seen that M-join-less runs more slowly than our two algorithms in any situation, this is because the computation of clique instances used for computing PI values is more expensive than that of intersection operations. Thus, in the following sections, we only discuss PBA and PTBA without M-join-less.

From Table 2, it can be seen that in both sparse and dense data sets, PBA shows better than PTBA, while in dense* data set with a very dense distribution, PTBA performs much better than PBA. The reason is that PTBA is a top-down method, and in dense* data set, the average size of prevalent co-locations is long and a long-size maximal co-locations may save much time on the identification of its subsets, while PBA is a bottom-up method that it needs more time to identify a long maximal co-location. Note that in PTBA algorithm, a denser data set can cause a high percentage of $T_{pruning}$. It also can be seen in Table 2 that T_{gen_mspcp} and $T_{pruning}$ consist of the main costs than any other costs. That is, the different methods for identifying candidates are the core distinction between two algorithms.

5.3 Comparison on expected costs in identifying candidates

From Table 2, it is hard to declare whether one of the two algorithms is better than the other one. Thus, in order to demonstrate the relationship of running time between the two algorithms, the expected costs are compared to identify candidates in PBA with those in PTBA.

Table 2 Comparison of computational complexity factors (factor %)

Method	M-join-less		PTBA			PBA		
data type	sparse	dense	sparse	dense	dense*	sparse	dense	dense*
$T_{gen\_2\_prev\_col}$	6.12	0.6	0.2	0.001	3.40	16.25	0.9	1.87
T_{gen_candi}	2.58	3.85	1.96	0.29	0.44	–	–	–
$T_{gen\_candi\text{-}tree}$	–	–	6.49	0.74	0.74	–	–	–
$T_{gen\_2\text{-}non\_prev\_col}$	–	–	–	–	–	2.52	0.001	0.001
$T_{order\_features}$	–	–	–	–	–	13.89	0.002	0.001
$T_{Part\_1+Part\_2}$	–	–	–	–	–	1.77	0.002	0.002
$T_{Pruning}$	–	–	3.12	39.21	61.53	8.59	4.16	0.26
T_{gen_mspcp}	91.29	95.64	88.19	59.76	33.88	56.89	94.94	97.88
Total execution time(Sec)	3.03	108.36	0.65	102.17	1.32	0.48	7.23	9.17

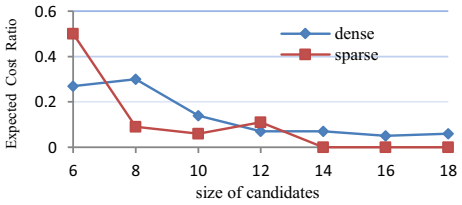


Figure 7 Comparison of expected cost in identifying candidates

A dense data set where the spatial frame size D is set as 1000 and a sparse data set where D is set as 5000 are used to evaluate the expected performance ratio. For each data set, all k-size candidates are selected where k is set as 6, 8, 10, 12, 14, 16 and 18, respectively. For each k-size candidates, the average execute time on different *min_sprev* is recorded for each algorithm where *min_sprev* is set as 0.2, 0.4, 0.6 and 0.8, respectively. Thus, the y-axis shows the ratio of average execute time of PBA to the average execute time of PTBA. As shown in Figure 7, the expected cost ratio decreases with the increase of the size of candidates, especially in sparse data sets. Meanwhile, it should be noted that the experimental results are basically consistent with the analysis in Section 4.2.

5.4 Comparison of candidate pruning ratios

We studied the effect of candidate pruning ratio with overlap ratio, which controls the false MSPCP-candidates generated from our 2-size SPCPs. In order to make our results more efficient, we reduced the range D and *min_sprev* in order to make sure that the candidate sets are dense enough to perform our experiments.

As Figure 8 shows, with the overlap ratio increases, the pruning ratio also increases in both PTBA and PBA, but PTBA can do better pruning than PBA although PBA has a good performance too. This is explained by the fact that when all the 2-size patterns are all prevalent, the PTBA has $2\text{^}|F| - (|F| + 1)$ candidates while PBA only has $|F|$ candidates, and if the longest candidate from prefix-tree generated in PTBA is prevalent, it will prune the whole tree because the remaining candidates are subsets of this candidate. We can also see that the candidate pruning ratio of PTBA is over 90% when the degree of overlap is bigger than 15%, and that of PBA is also over 90% when the degree of overlap is 20%.

5.5 Effects of the Parameter Clumpy

We examined the effects of the parameter *clumpy* for PTBA and PBA. *Clumpy* shows the number of cluster instances generated in the same neighborhood area. The bigger the clumpy degree is, the more cluster instances gather in the same neighborhood area. As Figure 9 shows,

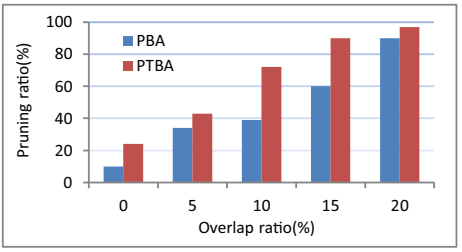


Figure 8 Comparison of candidate pruning ratio

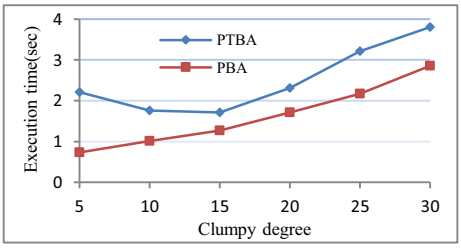


Figure 9 Effects of the parameter *clumpy*

as *clumpy* increases, PTBA and PBA all increase very slowly. In fact, their $T_{gen\_max\_prev\_coloc}$ is almost the same for different *clumpy* values. Only $T_{gen\_2\text{-}prev\_coloc}$ is affected as the clumpy degree increases. So, both PTBA and PBA have good robustness with respect to the parameter *clumpy*.

5.6 Scalability tests

We examined the scalability of PTBA and PBA with several workloads on execution time (TPTBA and TPBA) and space cost (SPTBA and SPBA), e.g. different numbers of instances, numbers of features, neighbor distance thresholds, and sub-prevalence thresholds.

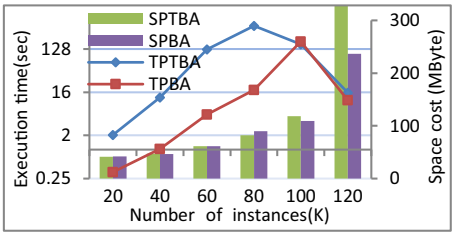
Effect of the number of instances First, the effect of the number of instances was compared between PTBA and PBA. In order to ensure that each data set generates around 10-size prevalent patterns, *P* (The average number of 2-size instances in 2-size core co-locations, see Table 1) was increased with the increase of the number of instances. As shown in Figure 10a, when the number of instances is fewer, PBA performs better than PTBA, but as the data set grows denser, PBA using a bottom-up way costs more time on calculation while PTBA adopting a top-down manner has a drop after the point of 80 k (80000) instances. As the number of instances increases (more than 100 K), a dense* data set is formed where the longest candidate pattern is sub-prevalent, which makes PTBA and PBA terminate soon and spend more on generating 2-size sub-prevalent patterns. Because PTBA costs more time on creating the prefix tree than PBA costs on Partition_1 and Partition_2, PTBA costs a little more than PBA. The space costs of both algorithms increase as the increase of the number of instances, and the main reason why PTBA costs much more than PBA when the number of instances over 100 K is that the data set gets dense enough that PTBA may generate many long-size non-prevalent candidates being checked while PBA can discard them by checking lower-size subsets.

Effect of the number of features Second, the performance of PTBA and PBA was compared with the number of features. As shown in Figure 10b, as the increase of features, PTBA shows a slow decrease while PBA shows a rapid increase. The main reason is that when the number of instances is fixed, an increase of the number of features may cause a decrease number of instances per feature, which in turn causes the size of result co-locations decrease. Thus, the shorter-size candidates may cause PBA cost more on Partition_2 from a larger number of non-prevalent 2-size patterns, but makes PTBA efficient because of the decrease of the number of instances per 2-size pattern. The space costs of the two algorithms are much the same because the data set is sparse and the average size of result co-locations and the average number of row-instances per co-location is quite low, which causes that the main space cost of both

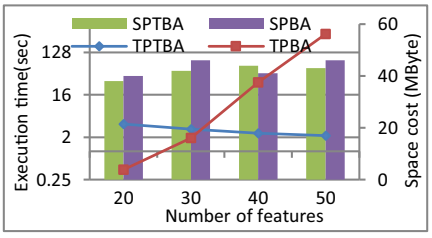
algorithms is the storage of row-instances. In sparse data sets, PBA may cost more space than PTBA mainly because of the recursive process. It needs to be said that when $N >$ 300,000 and the number of features reaches 40, a memory overflow was occurred in implementation of PTBA.

Effect of neighbor distances The effect of different neighbor distances was examined in the third experiment. As Figure 10c shows, both PTBA and PBA first have an increase then decrease. The main reason of increase is that a larger neighbor distance causes larger neighborhood area that causes a larger number of 2-size co-location instances. During this process of the increase of neighbor distances, PBA performs better than PTBA. But when the distance is large enough (e.g. more than 100 in Figure 10c), the data set gets dense enough that the number and the average size of maximal sub-prevalent co-locations increase. These increases make PTBA which adopts a top-bottom way to check a long candidate more efficient than PBA using a bottom-up manner because the high probability of the longest candidate being sub-prevalent. While the space costs of the two algorithms increase as the increase of neighbor distances, because a denser data set makes longer size candidates and larger count of row-instances, and in a dense data, PBA performs much better than PTBA.

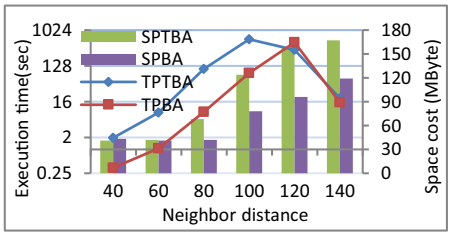
Effect of sub-prevalence threshold Finally, the performance effect of different sub-prevalence threshold *min_sprev* was examined. Overall, an increase of sub-prevalence threshold causes decreases of the execution time for both algorithms, as shown in Figure 10d. However, the PBA algorithm drops slowly and has better time performance than PTBA algorithm as the increase of *min_sprev*. With the increase of thresholds, the space costs decrease because a higher threshold may cause fewer candidates with less size and row-instances. When threshold is low, PTBA will generate much more long candidates than PBA, thus, PBA performs well than PTBA.



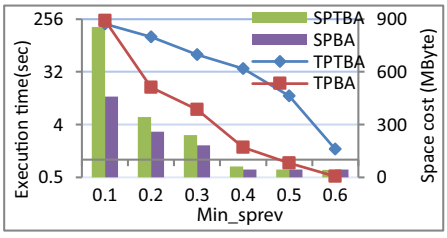
(a) By number of instances



(b) By number of features



(c) By neighbor distances



(d) By sub-prevalence thresholds

Figure 10 Evaluation and comparison of the two algorithms on execution time and space cost

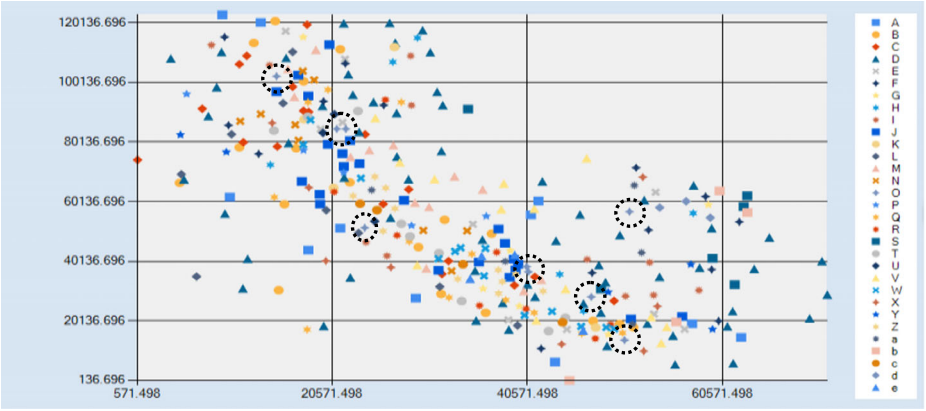


Figure 11 The distribution of a certain plant data

5.7 Evaluation with Real Data Sets

In this section, a real data set of the "Three Parallel Rivers of Yunnan Protected Areas" is used to test our algorithms. The distribution of this data set can be observed from Figure 11, where X and Y represent the location of each instance respectively. It can be observed that the data set forms a zonal distribution, which is common in natural ecological studies. This data set contains 31 features (plants) and 336 instances within a rectangular spatial area of size 125 km × 720 km.

First, mining result was tested to show the differences between MSPCPs (Maximal Sub-prevalent Co-location Patterns) and MPCPs (Maximal Prevalent co-location Patterns) using the real data set. The neighbor distance threshold d is set as 12 km, and prevalence thresholds *min_sprev* and *min_prev* are set as 0.3. Table 3 shows the number of MSPCPs / MPCPs with different sizes.

From Table 3, it can be seen that the number of MSPCPs with longer-size is more than that of MPCPs with the same parameters. This is because for MPCPs, each row-instance of a MPCP must form a clique relationship. While for MSPCPs, each instance of a MSPCP may not form a clique relationship. Thus, for a pattern c, the number of its instances with clique relationship is no more than that with star partition relationship, in the other words, for the same co-location c, $\mathrm{PI}(c) \leq \mathrm{SPI}(c)$, thus, long-size patterns are more likely to be sub-prevalent. It also can be found in Table 3 that the percentage of long-size patterns with size over 7 in MSPCPs is 15.5%, while in MPCPs is only 4.1%. It is significance that long patterns containing more interesting information can be more likely to draw users' attentions.

Table 3 Mining Results on the Plant Data Set in Figure 11

size	N. of MSPCPs	N. of MPCPs	size	N. of MSPCPs	N. of MPCPs
2	5	15	7	23	31
3	41	70	8	35	9
4	64	115	9	10	5
5	95	98	10	7	3
6	72	73	11	3	–

Let's take some mined results as examples to do further analysis. Given three mined 11-size MSPCPs are {B, G, J, O, Q, R, T, V, W, c, e}, {G, J, O, Q, R, T, V, W, a, c, e} and {G, H, J, O, Q, R, V, W, a, c, e}, whilst three mined 10-size MPCPs are {G, J, O, R, T, V, W, a, c, e}, {G, J, Q, R, T, V, W, a, c, e} and {J, O, Q, R, T, V, W, a, c, e}. We note that 11-size MSPCPs are supersets of 10-size MPCPs. Let us consider one of the the rare plant "O" with dashed circles in Figure 11, The reason why it occurs in three mined 11-size MSPCPs is that it's basically distributed in an axial line which is the focus area of growing plants, the same cases to the plants "G" and "Q".

Second, another large scale vegetation distribution data set in the "Three Parallel Rivers of Yunnan Protected Area" was used to examine the efficiency of the PBA method, where the number of features is 15, and the number of instances is 487,857. *min_sprev* was set as 0.3, and the neighborhood distance thresholds are set as 1000 m, 3000 m, and 5000 m, respectively. As a result, the running time of PBA is 148.26(s), 1967.84(s), and 7906.74 (s), respectively.

6 Related Works

The problem of discovering association rules based on spatial relationships was firstly proposed by Koperski and Han [9]. This work discovers the subsets of spatial features frequently associated with a specific *reference feature*, e.g., mines. A top-down, progressive refinement method to discover all rules from the transactions was presented in [9], and Wang et al. [24] proposed a novel method based on the partition of spatial relationships for mining multilevel spatial association rules from the transactions. In their method, the introduction of an equivalence partition tree method makes the discovery of rules efficient.

Morimoto [14] adopted a *support count* measure for discovering frequent co-located features sets. This approach uses a space partitioning and non-overlap grouping scheme for identifying co-located instances. However, the explicit space partitioning approach may miss co-location instances across partitions. Shekhar and Huang [6, 16] proposed the *minimum participation ratio* based on *clique instances* to measure the frequency of a co-location pattern. This is a statistically meaningful interest measure for spatial co-location patterns. Thus, many mining algorithms were proposed [6–8, 18, 26, 27, 31, 32] using this interest measure. Specifically, [6] presented a classic *join-based* mining algorithm. The core idea of this approach is to find size k clique instances by joining the instances of its size k-1 subset co-locations where the first k-2 objects are common and then checking the neighbor relationships between the k-1th objects. [32] proposed a star-neighborhood-based *join-less* algorithm. The join-less method uses an instance-lookup scheme instead of the instance join operation used in the join-based method for identifying clique instances. [18] put forward a *CPI-tree* algorithm which uses a tree-structure to store the neighbor relationships between spatial instances, and the clique instances of candidate patterns can be quickly generated by CPI-tree.

In terms of uncertain data, [11] utilized a probability density function to describe the uncertainty of spatial instances' locations, defined the expected distance between instances and proposed the UJoin-based algorithm. Based on the concept of semantic proximity neighborhoods under the fuzzy equivalent classes of instances, [21] studied the problem of discovering co-location patterns from interval data. [23] considered the uncertainty of spatial instances' existence, and defined a prevalence probability and expected participation index based on a possible world model, and then provided exact and approximation algorithms that mine probabilistic prevalent co-location patterns. [15] proposed two new kinds of co-location pattern mining for *fuzzy objects*,

single co-location pattern mining (SCP) and range co-location pattern mining (RCP), to mining co-location patterns at a membership threshold or within a membership range.

To deal with the situation where there exists a rare feature in the data sets (i.e. the number of instances with this feature is significantly smaller than those with other features), [5] proposed a *maximal participation ratio* (maxPR) interest measure and a *maxPrune* algorithm based on a weak monotonicity property of the maxPR measure, and [4] defined a *minimum weighted participation ratio* interest measure and gave a mining algorithm based on this ratio, which can not only mine the prevalent co-location patterns with rare features, but also exclude the non-prevalent patterns.

In some cases, the co-location patterns are not prevalent globally, or some low participation index patterns are still prevalent in a specific region. Therefore, regional co-location pattern mining has become a research focus [1, 2]. The huge size of prevalent co-location patterns does not help the user to easily retrieve relevant information. Such observation leads to various definitions of redundancy in order to limit the number of spatial prevalent co-location patterns [17, 19, 25, 28–30]. Recently domain-driven spatial co-location pattern discovery has been attracting more researchers [3, 12, 13, 22].

Although there were a lot of researches about co-location pattern mining, the problem about sub-prevalent co-location pattern mining was taken the first step by us in [20]. We enlarge that work here by adding more theoretical and experimental analysis.

Our work on the interest measure lies between the *reference object* model [9] and the *minimum participation ratio* measure [6, 16]. Without specific reference feature, directly applying the model of [9] may not capture some co-locations found by the techniques in this paper, whilst the measure in [6, 16] may miss some reasonable and useful co-located patterns in practical applications because of the requirement of clique instances. In the designs for mining MSPCPs, the idea of using the prefix-tree structure comes from the work presented in [25]; the idea of pruning breadth-first in candidate space search tree can be traced back to [28, 30]. In the literature we have found no mention of our core pattern method based on partitions.

7 Conclusions

In traditional spatial co-location pattern mining, instances forming a clique relationship are used as a row-instance of a co-location pattern. This definition is too strict and may lose some significance of the co-located features in practice. To solve this problem, new kinds of co-location patterns which have more meaningful significance for co-located features are proposed in this paper. Thanks to the downward inclusion property of our proposed star participation ratio (SPR) and star participation index (SPI), it makes the discovery of sub-prevalent maximal co-location pattern mining feasible.

In order to discover maximal sub-prevalent co-locations, two novel algorithms PBA and PTBA were proposed in this paper, and extent experiments were conducted to evaluate the performances of them. Empirical evaluation shows that our two algorithms complement each other for different situations, i.e., PBA is more suitable to sparse data sets while PTBA performs well in dense data sets. Overall, our proposed two algorithms are more efficient than M-join-less algorithm in any case.

For the future plan, we mean to extend the maximal sub-prevalent co-location mining to other types of spatial data, e.g. spatio-temporal data and uncertain data.

Acknowledgements This work is supported by the National Natural Science Foundation of China (No. 61472346, No. 61662086 and No. 61762090), the Natural Science Foundation of Yunnan Province (No. 2016FA026), the Program for Young and Middle-aged Skeleton Teachers of Yunnan University (No. WX069051), and the Project of Innovation Research Team of Yunnan Province (2018HC019).

References

1. Akbari, M., Samadzadegan, F., Weibel, R.: A generic regional spatio-temporal co-occurrence pattern mining model: a case study for air pollution. J. Geogr. Syst. **17**, 249–274 (2015)
2. Celik, M., Kang, J.M., Shekhar, S.: Zonal co-location pattern discovery with dynamic parameters. In: Proceedings of the 7th IEEE International Conference on Data Mining, pp. 433–438, IEEE (2007)
3. Fang, Y., Wang, L., Wang, X., Zhou, L.: Mining co-location patterns with dominant features, In: Proceedings of the 18th International Conference on Web Information Systems Engineering (WISE 2017), LNCS 10569, pp. 183–198 IEEE (2017)
4. Feng, L., Wang, L., Gao, S.: A new approach of mining co-location patterns in spatial datasets with rare features. Journal of Nanjing University (Natural Sciences). **48**(1), 99–107 (2012)
5. Huang, Y., Pei, J., Xiong, H.: Mining co-location patterns with rare events from spatial data sets. GeoInformatica. **10**(3), 239–260 (2006)
6. Huang, Y., Shekhar, S., Xiong, H.: Discovering co-location patterns from spatial data sets: A General Approach. IEEE Trans. Knowl. and Data Eng. (TKDE). **16**(12), 1472–1485 (2004)
7. Huang, Y., Zhang, P.: On the relationships between clustering and spatial co-location pattern mining. In: Proceedings of the 18th IEEE Int. Conf. on Tools with Artificial Intelligence (ICTAI 06). pp. 513–522. IEEE (2006)
8. Huang, Y., Zhang, L., Yu, P.: Can we apply projection based frequent pattern mining paradigm to spatial co-location mining? In: Proceedings of the 9th Pacific-Asia Conf. on Knowledge Discovery and Data Mining (PAKDD 2005). pp. 719–725. IEEE (2005)
9. Koperski, K., Han, J.: Discovery of spatial association rules in geographic information databases. In: Proceedings of the Fourth International Symposium, Large Spatial Databases (SSD'95). pp. 47–66. (1995)
10. Li, J., Adilmagambetov, A., Jabbar, M., et al.: On discovering co-location patterns in datasets: a case study of pollutants and child cancers. Geoinformatica. (2016). https://doi.org/10.1007/s10707-016-0254-1
11. Liu, Z., Huang, Y.: Mining co-locations under uncertainty, In: Proceedings of the 13th International Symposium on Spatial and Temporal Databases (SSTD), pp. 429–446, IEEE (2013)
12. Lu, J., Wang, L., Fang, Y., Zhao, J.: Mining strong symbiotic patterns hidden in spatial prevalent co-location patterns. Knowledge-Based Systems (KBS). **146**(2018), 190–202 (2018)
13. Lu, J., Wang, L., Fang, Y., et al.: Mining competitive pairs hidden in co-location patterns from dynamic spatial databases, In: Proceedings of the 21st Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining (PAKDD), pp. 467–480, IEEE (2017)
14. Morimoto, Y.: Mining frequent neighboring class sets in spatial databases. In: Proceedings of the 7th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), pp. 353–358. ACM (2001)
15. Ouyang, Z., Wang, L., Wu, P.: Spatial co-location pattern discovery from fuzzy objects. International Journal of Artificial Intelligence Tools, 26 (2017) 1750003 (20 pages) (2017)
16. Shekhar, S., Huang, Y.: Co-location rules mining: A summary of results. In: Proceedings of the IEEE International Symposium on Spatial and Temporal Databases (SSTD), pp. 236–256 (2001)
17. Wang, L., Bao, X., Chen, H., Cao, L.: Effective lossless condensed representation and discovery of spatial co-location patterns. Inf. Sci. **436-437**(2018), 197–213 (2018)
18. Wang, L., Bao, Y., Lu, J., Yip, J.: A new join-less approach for co-location pattern mining. In: Proceedings of the IEEE 8th International Conference on Computer and Information Technology, pp. 197–202. IEEE (2008)
19. Wang, L., Bao, X., Zhou, L.: Redundancy reduction for prevalent co-location patterns. IEEE Trans. Knowl. Data Eng. (TKDE). **30**(1), 142–155 (2018)
20. Wang, L., Bao, X., Zhou, L., Chen, H.: Maximal sub-prevalent co-location patterns and efficient mining algorithms, In: Proceedings of the 18th international conference on Web Information Systems Engineering (WISE), LNCS 10569, pp. 199–214 IEEE (2017)

21. Wang, L., Chen, H., Zhao, L., Zhou, L.: Efficiently mining co-location rules on interval data. In: Proceedings of The 6th International Conference on Advanced Data Mining and Applications, LNCS 6440, pp. 477–488 Springer (2010)
22. Wang, L., Jiang, W., Chen, H., Fang, Y.: Efficiently mining high utility co-location patterns from spatial data sets with instance-specific utilities. In: the 22nd Int. Conf. on Database Systems for Advanced Applications (DASFAA). LNCS. **10178**, 458–474 (2017)
23. Wang, L., Wu, P., Chen, H.: Finding probabilistic prevalent co-locations in spatially uncertain data sets. IEEE Trans. Knowl. Data Eng. (TKDE). **25**(4), 790–804 (2013)
24. Wang, L., Xie, K., Chen, T., Ma, X.: Efficient discovery of multilevel spatial association rules using partitions. Inf. Softw. Technol. **47**(2005), 829–840 (2005)
25. Wang, L., Zhou, L., Lu, J., Yip, J.: An order-clique-based approach for mining maximal co-locations. Inf. Sci. **179**(2009), 3370–3382 (2009)
26. Xiao, X., Xie, X., Luo, Q., Ma, W.: Density based co-location pattern discovery. In: Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS), pp. 11–20 ACM (2008)
27. Yao, X., Chen, L., Peng, L., Chi, T.: A co-location pattern-mining algorithm with a density-weighted distance thresholding consideration. Inf. Sci. **396**, 144–161 (2017) 2017)
28. Yoo, J.S., Bow, M.: Mining maximal co-located event sets. In: Proceedings of the 15th Pacific-Asia International Conference on Knowledge Discovery and Data Mining (PAKDD), LNCS 6634, pp. 351–362 (2011)
29. Yoo, J.S., Bow, M.: Mining top-k closed co-location patterns. In: Proceedings of IEEE International Conference on Spatial Data Mining and Geographical Knowledge Services, pp.100–105 IEEE (2011)
30. Yoo, J.S., Bow, M.: Mining spatial co-location patterns: a different framework. Data Mining and Knowledge Discovery Journal. **24(1**, 159–194 (2012)
31. Yoo, J.S., Shekhar, S.: A partial join approach for mining co-location patterns. In: Proceedings of the 12th Annual ACM International Workshop on Geographic Information Systems, pp. 241–249 ACM (2004)
32. Yoo, J.S., Shekhar, S.: A Joinless Approach for Mining Spatial Co-location Patterns. IEEE Trans. Knowl. Data Eng. (TKDE). **18**(10), 1323–1337 (2006)