Redundancy Reduction for Prevalent Co-Location Patterns

Lizhen Wang, *Member, IEEE*, Xuguang Bao, and Lihua Zhou

Abstract—Spatial co-location pattern mining is an interesting and important task in spatial data mining which discovers the subsets of spatial features frequently observed together in nearby geographic space. However, the traditional framework of mining prevalent co-location patterns produces numerous redundant co-location patterns, which makes it hard for users to understand or apply. To address this issue, in this paper, we study the problem of reducing redundancy in a collection of prevalent co-location patterns by utilizing the spatial distribution information of co-location instances. We first introduce the concept of *semantic distance* between a co-location pattern and its super-patterns, and then define redundant co-locations by introducing the concept of *δ-covered*, where $\delta\,(0 \le \delta \le 1)$ is a coverage measure. We develop two algorithms *RRclosed* and *RRnull* to perform the redundancy reduction for prevalent co-location patterns. The former adopts the *post-mining* framework that is commonly used by existing redundancy reduction techniques, while the latter employs the *mine-and-reduce* framework that pushes redundancy reduction into the co-location mining process. Our performance studies on the synthetic and real-world data sets demonstrate that our method effectively reduces the size of the original collection of closed co-location patterns by about 50 percent. Furthermore, the RRnull method runs much faster than the related closed co-location pattern mining algorithm.

Index Terms—Spatial co-location pattern, redundancy, semantic distance, *δ-covered*

1 Introduction

Mining spatial co-location patterns is an interesting and important spatial data mining task with a broad range of applications including Earth science [1], public health [2], public transportation [3], environmental studies [4] and et al. In [5] for example, the extracted patterns in the public-service facilities of developed cities can be used to plan layouts or arrange new facilities in cities.

A spatial *co-location pattern* is a group of spatial features whose instances are frequently located close to each other [6]. Examples of spatial co-location patterns include symbiotic species, e.g., West Nile Virus and stagnant water sources in public health, and interdependent incidents, e.g., traffic jams, car accidents, ambulances and police in public transportation.

The traditional framework of spatial co-location pattern mining uses the frequencies of a set of spatial features participating in a co-location pattern to measure the prevalence (known as *participation index* [6], or *PI* for short) and requires a user-specified minimum PI threshold to find interesting co-location patterns. The meaning of PI is that wherever a feature in a co-location pattern *c* is observed, all other features in *c* can be observed in its neighborhood with a probability of at least *PI*(*c*). Similar to the support metric in frequent itemset mining, the PI metric satisfies the *anti-monotonicity property*. That is, if a spatial co-location pattern is prevalent with respect to a threshold of PI, then all of its subsets will be discovered as prevalent co-location patterns. Traditional frameworks generate numerous redundant co-location patterns which jeopardize the usability of the technique, as it then demands great effort to discern or understand the discovered knowledge.

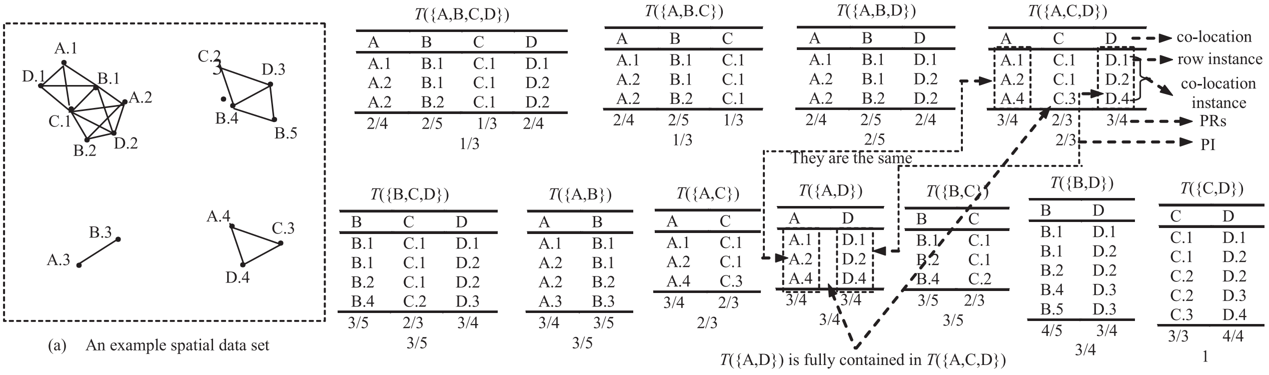
Two major approaches have been developed to aid the user: *lossless* and *lossy* redundancy reduction. The former, using *closed prevalent co-locations* (CPC) [7] (a prevalent co-location c is closed if there is no co-location c' such that $c \subset c'$ and $\mathrm{PI}(c) = \mathrm{PI}(c')$), concentrates too much on the PI information of co-locations so that its redundancy reduction power is quite limited. The latter, using *maximal prevalent co-locations* (MPC) [8], [9] (a prevalent co-location c is maximal if there is no prevalent co-location c' such that $c \subset c'$), may significantly reduce the number of co-location patterns, but it loses PI information from most of the co-locations, leaving difficulties for the user. This paper presents a new, improved, redundancy reduction framework for detecting spatial prevalent co-location patterns, utilizing the spatial distributed information of co-location instances whilst retaining some useful features of the non-redundant co-location sets

An explanatory example is shown below:

Example 1. Fig. 1a shows an example spatial data set, where instances of four spatial features, A, B, C and D, are denoted by the feature type and a numeric id value, e.g., A.1, and edges connecting instances indicate spatial

- *The authors are with the Department of Computer Science and Engineering, School of Information Science and Engineering, Yunnan University, Kunming 650221, China.*
E-mail: lzhwang2005@126.com, bbaaooxx@163.com, lhzhou@ynu.edu.cn.

Manuscript received 28 Feb. 2016; revised 28 Aug. 2017; accepted 25 Sept. 2017. Date of publication 4 Oct. 2017; date of current version 5 Dec. 2017.
(Corresponding author: Lizhen Wang.)
Recommended for acceptance by H. Lee.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TKDE.2017.2759110



(a) An example spatial data set

$T(\{A,B,C,D\})$

A	B	C	D
A.1	B.1	C.1	D.1
A.2	B.1	C.1	D.2
A.2	B.2	C.1	D.2
2/4	2/5	1/3	2/4

1/3

$T(\{A,B.C\})$

A	B	C
A.1	B.1	C.1
A.2	B.1	C.1
A.2	B.2	C.1
2/4	2/5	1/3

1/3

$T(\{A,B,D\})$

A	B	D
A.1	B.1	D.1
A.2	B.1	D.2
A.2	B.2	D.2
2/4	2/5	2/4

2/5

$T(\{A,C,D\})$

A	C	D
A.1	C.1	D.1
A.2	C.1	D.2
A.4	C.3	D.4
3/4	2/3	3/4

2/3

$T(\{B,C,D\})$

B	C	D
B.1	C.1	D.1
B.1	C.1	D.2
B.2	C.1	D.2
B.4	C.2	D.3
3/5	2/3	3/4

3/5

$T(\{A,B\})$

A	B
A.1	B.1
A.2	B.1
A.2	B.2
A.3	B.3
3/4	3/5

3/5

$T(\{A,C\})$

A	C
A.1	C.1
A.2	C.1
A.4	C.3
3/4	2/3

2/3

$T(\{A,D\})$

A	D
A.1	D.1
A.2	D.2
A.4	D.4
3/4	3/4

3/4

$T(\{B,C\})$

B	C
B.1	C.1
B.2	C.1
B.4	C.2
3/5	2/3

3/5

$T(\{B,D\})$

B	D
B.1	D.1
B.1	D.2
B.2	D.2
B.4	D.3
B.5	D.3
4/5	3/4

3/4

$T(\{C,D\})$

C	D
C.1	D.1
C.1	D.2
C.2	D.2
C.2	D.3
C.3	D.4
3/3	4/4

1

(b) The co-location instances, PR values and PI values of possible co-locations in data set of Fig. 1a

Fig. 1. An explanatory example.

neighboring relationships. Fig. 1b lists the co-location instances, the PRs and the PIs of all possible co-locations in the data set (the definitions of co-location instance, PR and PI are provided in Section 2.1). If a minimum PI threshold is 0.3, MPC mining will only report the 4-size co-location pattern {A, B, C, D}. In contrast, the result of CPC mining will be {{A, B, C, D}, {A, B, D}, {A, C, D}, {B, C, D}, {A, B}, {A, D}, {B, D}, {C, D}}. However, we observe that each co-location in the set {{A, B, D}, {A, C, D}, {B, C, D}, {A, B}, {A, D}, {B, D}, {C, D}} is significantly different with respect to their PI values from {A, B, C, D}. Additionally, CPC mining is too prolific, as we note that the co-location instance T({A, D}) of {A, D} is fully contained in co-location instance T({A, C, D}) of its super-pattern {A, C, D} (as shown in the dotted boxes in T({A, D}) and T({A, C, D})). We say {A, D} is *covered* by {A, C, D} with respect to the distributed information of co-location instances (see Definition 7). Similarly, co-locations {A, B, D} and {B, D} are covered by {A, B, C, D} and {B, C, D} respectively, and the co-location instance information of {A, C, D} and {B, C, D} covers that of {C, D}. The high-quality non-redundant result is {{A, B, C, D}, {A, C, D}, {B, C, D}, {A, B}} for this example data set.

A strategy for improving the redundancy reduction power in CPCs is to identify redundant co-locations according to certain similarity measures. However, there are three crucial questions which need to be answered: 1) how to measure the similarity between the co-location patterns, 2) how to efficiently and completely eliminate redundant co-locations and, for research purposes, 3) how to estimate the redundancy degree of co-location patterns.

This paper is an attempt to offer answers to these questions. First, we propose a *semantic distance* metric between a co-location and its super-patterns, and show it is a sub-valid distance metric. Second, we define a concept of ***δ**-covered* to estimate the redundancy degree of co-location patterns. Lastly, we propose two algorithms: *RRclosed*, which follows existing redundancy reduction techniques to adopt the post-mining framework that reduces redundant co-locations from the set of CPCs; *RRnull*, which employs a *mine-and-reduce* framework to discover non-redundant results directly from the spatial data sets and runs much faster than the CPC mining algorithm in [7], itself a very fast CPC mining method. Our performance study shows that the introduction of ***δ**-covered* can effectively reduce the number of CPCs. In addition, users can control the redundancy reduction power by adjusting the coverage measure $\delta\,(0 \leq \delta \leq 1)$.

The remainder of the paper is organized as follows. In Section 2, we give the related definitions for prevalent co-location patterns' redundancy reduction. Section 3 presents the *RRclosed* method, and the *RRnull* method is proposed in Section 4. Our performance study is presented in Section 5. The related work is discussed in Sections 6, and 7 concludes.

2 Problem Definition

In this section, we first review the basic concept of co-location patterns. Then, we introduce a semantic distance metric to measure the similarity between the co-location patterns, and then the prevalent co-location pattern redundancy reduction problem is defined formally, based on a new *δ-covered* concept, where $\delta\,(0 \leq \delta \leq 1)$ is a coverage measure.

2.1 Co-Location Patterns

A spatial feature f_i represents a specific kind of thing in an area. For example, a plant species is a feature. An occurrence of f_i at a location is called an *instance* of f_i, e.g., a plant of certain species is an instance. We use the *spatial neighbor relationship NR* to describe the relationships between instances. When a euclidean metric is used for the spatial neighbor relationship *NR*, two instances are neighbors if the distance between them is not greater than a given distance threshold d.

Suppose F is a set of spatial features, and S is a set of their observed instances. Fig. 1a shows an example spatial data set, where $F = \{\mathrm{A, B, C, D}\}$ with each instance denoted by the feature type and a numeric id value, e.g., A.1. Edges among the instances indicate neighboring relationships under *NR*. Feature type A has four instances, B has five instances, C has three instances, and D has four instances in the data set. A *Spatial co-location pattern c* is a subset of the features' set F, $c \subseteq F$, whose instances form cliques frequently under *NR* (i.e., are frequently neighbors to each other). The *size* of *c* is the number of features in *c*.

We now review the measures used to characterize the interest of a co-location pattern. The detailed derivations can be obtained from [1], [2], [3], [4], [5], [6], [7], [8], [9], [10].

Definition 1 (Row Instance and Co-location Instance). *Given a collection of instances S of a set of spatial feature* F *and a co-location pattern c, a set of instances* $I \subseteq S$ *is called a* row instance *of c, if 1) I contains instances of all features in c and no proper subset of I does so, and 2) the instances in I form clique relations under the NR neighbor relationship. The* co-location instance, *T(c), of c is the collection of all row instances of c.*

For example, in Fig. 1a, {A.1, C.1, D.1} is a row instance of the co-location pattern {A, C, D} but {A.2, C.1, D.1} is not, and the co-location instance $T(\{A, C, D\})$ of {A, C, D} is {{A.1, C.1, D.1}, {A.2, C.1, D.2}, {A.4, C.3, D.4}}.

Definition 2 (Participation ratio). *Given a co-location pattern c, the* participation ratio *of a feature $f_i \in c$ 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 3 (Participation index). *The* participation index *of a co-location pattern c is defined as*

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

For example, in Fig. 1, T({A, C, D}) = {{ A.1, C.1, D.1}, {A.2, C.1, D.2}, {A.4, C.3, D.4}}. The participation ratio of feature A in {A, C, D}, *PR*({A, C, D}, A), is 3/4 since only A.1, A.2 and A.4 among the four instances of A are involved in T({A, C, D}). In the same way, *PR*({A, C, D}, C) is 2/3 and *PR*({A, C, D}, D) is 3/4. The participation index of {A, C, D}, *PI*({A, C, D}), is *min*{ *PR*({A, C, D}, A), *PR*({A, C, D}, C), *PR*({A, C, D}, D)} = 2/3.

Definition 4 (Prevalent co-location pattern). *Given a user-specified PI threshold M, a co-location pattern c is a* prevalent co-location pattern *if $PI(c) \geq M$.*

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].

Definition 5 (Closed prevalent co-location). *A prevalent co-location c is closed if there is no co-location c' such that $c \subset c'$ and $PI(c) = PI(c')$.*

The introduction of CPC mining creates a lossless redundancy reduction method [7], which can not only infer the original collection of prevalent co-locations but also their PI values.

2.2 Semantic Distance

To improve the redundancy reduction power in the set of CPCs, we introduce a semantic distance metric to measure the similarity between two CPCs based on their co-location instances, which contain the neighbor relationship information of their spatial instances.

Definition 6 (Semantic Distance, SD). *Let c and c' be two CPCs, and $c \subset c'$. The* semantic distance *between c and c' is defined as:*

$$SD(c, c') = \min_{f_i \in c}\left\{\left(1 - \frac{|\Pi_{f_i} T(c')|}{|\Pi_{f_i} T(c)|}\right)\right\}, \quad (3)$$

where $\Pi_{f_i} T(c)$ is the set of distinct instances of f_i in $T(c)$, and $T(c)$ is the co-location instance of c.

Let us apply the *SD* measure to the co-location patterns in Fig. 1 to see whether it reasonably reflects the distance between co-locations in term of redundancy. Firstly, we consider $c = \{A, D\}$ and $c' = \{A, C, D\}$. According to Definition 6, $|\prod_A T(c)| = |\{A.1,\ A.2,\ A.4\}| = 3$, $|\prod_D T(c)| = |\{D.1,\ D.2,\ D.4\}| = 3$, $|\prod_A T(c')| = |\{A.1,\ A.2,\ A.4\}| = 3$, $|\prod_D T(c')| = |\{D.1,\ D.2,\ D.4\}| = 3$, so $SD(c,\ c') = \min\{1 - 3/3,\ 1 - 3/3\} = 0$. This means the co-location instance information of {A, D}, which shows the prevalence information of a co-location pattern, is fully contained in that of {A, C, D} (i.e., when the instances of features A and D are observed in a neighborhood, the instance of feature C must occur in this neighborhood too). That is, {A, D} is a redundant co-location relative to {A, C, D}. Secondly, let us consider $c = \{A, B\}$ and $c' = \{A, B, C, D\}$. We can calculate $SD(c,\ c') = \min \{1 - 2/3,\ 1 - 2/3\} = 1/3$, which indicates that {A, B} have extra row instance distribution information relative to {A, B, C, D}. Finally, let us illustrate the meaning of the "min" in Eq. (3). We observe that $SD(\{B, D\}, \{B, C, D\}) = 0$ but $T(\{B, D\})$ has not been fully contained in $T(\{B, C, D\})$. In fact, there is no extra instance of feature D in $T(\{B, D\})$ which occurs relative to $T(\{B, C, D\})$. As far as *the distribution of co-location instances* is concerned, $T(\{B, D\})$ does not contain extra information relative to $T(\{B, C, D\})$ (Because D.3 in the row instance {B.5, D.3} has occured in row instance {B.4, C.2, D.3} of $T(\{B, C, D\})$, {B.5, D.3} is not a new distribution of row instances). Thus we see that the *SD* measure captures the redundancy power between a co-location and its super-pattern.

Theorem 1. *The semantic distance SD is a sub-valid distance metric, such that:*

1) $SD(c,\ c') \geq 0, \forall c \subset c'$
2) $SD(c,\ c') = 0, \forall c = c'$
3) *For* $\forall c \subset c' \subset c''$, *if* $\arg_f \max\{(\frac{|\Pi_f T(c')|}{|\Pi_f T(c)|}) :\ f \in c\} = \arg_f \max\{(\frac{|\Pi_f T(c'')|}{|\Pi_f T(c')|}) :\ f \in c'\}$ *then* $SD(c, c') + SD(c', c'') \geq SD(c, c'')$

Proof. By the definition of *SD*, it is easy to verify that the first two properties are true. We prove the third statement as below.

To simplify the presentation, we define the variables:

$$x = \max_{f \in c}\left\{\frac{|\Pi_f T(c')|}{|\Pi_f T(c)|}\right\}, y = \max_{f \in c'}\left\{\frac{|\Pi_f T(c'')|}{|\Pi_f T(c')|}\right\}$$
$$z = \max_{f \in c}\left\{\frac{|\Pi_f T(c'')|}{|\Pi_f T(c)|}\right\}$$

Plug in all the variables into the distance definition.

$$\begin{aligned} & SD(c,\ c') \ + \ SD(c',\ c'') \ \geq \ SD(c,\ c'') \\ & \Leftrightarrow \ (1 - \ x) \ + \ (1 - \ y) \ \geq \ (1 - \ z) \end{aligned} \quad (4)$$

If $c \subset c'$ then every row instance of c' contains a subset of instances which is a row instance of c. So $|\prod_f T(c)| \geq |\prod_f T(c')|$ for $f \in c$.

We can assume $x = b/a$ and $y = c/b$ in Eq. (4) based on the condition of the third statement. Therefore,

if $z = c/a$, we have $a \geq b \geq c$ due to $c \subset c' \subset c''$.

Then, $(1 - b/a) + (1 - c/b) = (1 - (b/a - ((b - c)/b)))$

$\geq (1 - (b/a - ((b - c)/a)))$ (by $a \geq b$)
$= (1 - c/a)$
So, Eq. (4) is true in this case.

Else, the values a and c in z are different from that in x and y. Since $z = \max_{f \in c}\{\frac{|\Pi_f T(c'')|}{|\Pi_f T(c)|}\}$, $1 - z \leq 1 - c/a$. Eq. (4) is still true.

Thus the third statement is true. □

Remark. 1) Although the SD can reasonably reflect the distance between co-locations in term of redundancy, it is sub-valid since there is a condition on the third statement in Theorem 1. 2) The SD can be extended to general prevalent co-locations excepting that, for a non-CPC c, there is a CPC c' such that $c \subset c'$ and $SD(c, c') = 0$. This is because from the condition $PI(c) = PI(c')$ of a non-CPC c, we can infer that there exists a feature f_i *in* c such that $PR(c, f_i) = PR(c', f_i)$, and therefore $SD(c, c') = 0$ holds.

2.3 δ-Covered

Based on the SD metric, we can define a covered relationship between co-locations, and then the concept of **δ**-*covered* is introduced further.

Definition 7 (Covered (or Non-covered)). *For a co-location c, if there exists (does not exist) a co-location c' such that $c \subset c'$ and $SD(c, c') = 0$. We say c is* covered *by c' (c is a* non-covered *co-location pattern).*

For example, co-location {A, D} is covered by {A, C, D} in the data set of Fig. 1a. If {A, C, D} has been in the set of CPCs, {A, D} is a *redundant* CPC with respect to the distribution of co-location instances. By contrast, the co-location pattern {A, B} in the data set of Fig. 1a is a non-covered co-location since its row instance {A.3, B.3} cannot be contained in any of its super-patterns.

Obviously, we can use the covered concept to prune redundant CPCs in the set of CPCs. For example, in Fig. 1, if $M = 0.3$, the CPC set is {{A, B, C, D}, {A, B, D}, {A, C, D}, {B, C, D}, {A, B}, {A, D}, {B, D}, {C, D}}. So CPCs {A, B, D}, {A, D} and {B, D} can be pruned since they are covered by {A, B, C, D}, {A, C, D} and {B, C, D} respectively.

In order to achieve a more succinct compression of the non-covered prevalent co-location patterns, we extend the concept of covered in two ways, as follows.

First, let c' be a set of the super-patterns of c, say $c' = \{c_1, c_2, \ldots, c_t\}$, and we then have the following extended concept of SD:

Definition 8 (Extended Semantic Distance, ESD). *Let c be a CPC and $\{c_1, c_2, \ldots, c_t\}\,(t \geq 1)$ be a set of CPCs, and $c \subset c_i\,(1 \leq i \leq t)$. The* extended semantic distance *between c and $\{c_1, c_2, \ldots, c_t\}$ is defined as:*

$$ESD(c, \{c_1, c_2, \ldots, c_t\}) = \min_{f_i \in c}\left\{\left(1 - \frac{|\bigcup_{j=1}^{t} \Pi_{f_i} T(c_j)|}{|\Pi_{f_i} T(c)|}\right)\right\}, \quad (5)$$

where $\Pi_{f_i} T(c)$ is the set of distinct instances of f_i in $T(c)$, and $T(c)$ is the co-location instance of c.

Naturally, the SD in Definition 7 can be extended to ESD, and then more co-locations in CPCs could be eliminated.

Example 2. In Fig. 1, if $M = 0.3$, for {C, D} in CPC set, there are two CPCs {A, C, D} and {B, C, D}, such that $SD(\{\mathrm{C}, \mathrm{D}\}, \{\{\mathrm{A}, \mathrm{C}, \mathrm{D}\}, \{\mathrm{B}, \mathrm{C}, \mathrm{D}\}\}) = \min\{1 - \frac{3}{3}, 1 - \frac{4}{4}\} = 0$. {C, D} is covered by its super-pattern set {{A, C, D}, {B, C, D}}. That is to say, if {A, C, D} and {B, C, D} are in the CPC set, {C, D} is deemed to be redundant and it should be eliminated so as to further reduce the number of non-covered co-locations in the set of CPCs.

Accordingly, for the data set of Fig. 1a, the CPC set is reduced to {{A, B, C, D}, {A, C, D}, {B, C, D}, {A, B}}. We regard this as the ideal non-redundancy result, because each pattern has extra information (i.e., an extra co-location instance, see Fig. 1a).

Second, we extend the *covered* concept to *δ-covered* to further reduce the number of non-covered co-locations in the set of CPCs by adjusting the coverage measure $\delta\,(0 \leq \delta \leq 1)$, where **$\delta$** is a user specified coverage measure threshold.

Definition 9 (δ-covered). *A co-location c is δ*-covered *by a set of co-locations $\{c_1, c_2, \ldots, c_t\}\,(t \geq 1)$ if $c \subset c_i\,(1 \leq i \leq t)$ and $ESD(c, \{c_1, c_2, \ldots, c_t\}) \leq \delta\,(0 \leq \delta \leq 1)$.*

2.4 The Problem Definition and Analysis

Based on the above discussion, the prevalent co-location redundancy reduction problem is formally defined as follows:

Definition 10 (Prevalent Co-location Redundancy Reduction Problem). *Given a spatial data set D (a collection of instances S of a set of spatial feature F), a minimum PI threshold M and a coverage measure δ, the* prevalent co-location redundancy reduction problem *is to find a minimal set of non-covered prevalent co-location patterns* **Ω**, *such that for any prevalent co-location pattern c in D, i.e., $PI(c) \geq M$, there exists a set of co-locations $\{c_1, c_2, \ldots, c_t\}\,(t \geq 1)$ in Ω s.t. $c \subset c_i$ and $ESD(c, \{c_1, c_2, \ldots, c_t\}) \leq \delta$.*

We note that the size of Ω is no less than the number of the maximal prevalent co-locations. This is because a MPC can only be covered by itself. On the other hand, the size of Ω is no larger than the number of the CPCs since a non-CPC must be covered by a CPC.

Theorem 2. *The δ-covered relationship is a limited partial order in the prevalent co-location set, such that:*

1) *c is δ-covered by c. (reflexivity)*
2) *if c is δ-covered by c' and c' is δ-covered by c, then $c = c'$. (anti-symmetry)*
3) *if c is covered by c' and c' is δ-covered by $\{c_1, c_2, \ldots, c_t\}$, and $f^* = \arg_f \max\{\frac{|\Pi_f T(c')|}{|\Pi_f T(c)|} : f \in c\} = \arg_f \max\{\frac{|\cup_{i=1}^{t} \Pi_f T(c_i)|}{|\Pi_f T(c')|} : f \in c'\}$then c must be δ-covered by $\{c_1, c_2, \ldots, c_t\}$. (limited-transitivity)*

Proof. By definition of *δ-covered*, it is easy to verify that the first two properties are true. We prove the third statement here.

According to the conditions of the third statement, we have

$$SD(c,c') = 1 - \max_{f\in c}\left\{\frac{|\Pi_f T(c')|}{|\Pi_f T(c)|}\right\} = 0 \text{ and}$$

$$ESD(c',\{c_1,\dots c_t\}) = 1 - \max_{f\in c'}\left\{\frac{|\bigcup_{i=1}^{t}\Pi_f T(c_i)|}{|\Pi_f T(c')|}\right\} \le \delta$$

$$\Rightarrow$$

$$ESD(c,\{c_1,...c_t\}) = 1 - \max_{f\in c}\left\{\frac{|\bigcup_{i=1}^{t}\Pi_f T(c_i)|}{|\Pi_f T(c)|}\right\}$$

$$= \left(1 - \frac{|\Pi_{f^*}T(c')|}{|\Pi_{f^*}T(c)|}\cdot\frac{|\bigcup_{i=1}^{t}\Pi_{f^*}T(c_i)|}{|\Pi_{f^*}T(c')|}\right) \le \delta$$

□

The semantic distance, a *sub-valid* distance metric, leads to the *limited-transitivity* expressed in Theorem 2, and it means that there may exist co-locations that do not satisfy *transitivity*. We call these *hard co-locations*.

Definition 11 (Hard co-location). *For a prevalent co-location c, if c is covered by its super-pattern c′ and c′ is δ-covered by its super-pattern set* $\{c_1, c_2, \dots, c_t\}$*, but c cannot be δ-covered by* $\{c_1, c_2, \dots, c_t\}$*, c is called a* hard co-location *of c′.*

For example, in Fig. 1, if the co-location instance of co-location pattern {A, B, C, D} became {{A.1, B.1, C.1, D.1}, {A.2, B.1, C.1, D.2}, {A.2, B.2, C.1, D.3}}, the co-location {B, C, D} was covered by {A, B, C, D} ($\delta = 0$). But the co-location {B, C} cannot be covered by {A, B, C, D}. Thus, {B, C} is a hard co-location pattern of {B, C, D}.

For a non-CPC c, there is a CPC c' such that and $c \subset c'$ and $SD(c, c') = 0$, so we have the following lemma, proof omitted.

Lemma 1. *Given a spatial data set D, a minimum PI threshold M and a coverage measure δ, if any co-location in the "closed + hard" co-location set is δ-covered by co-locations in Ω, then any prevalent co-location in D can be δ-covered by co-locations in Ω.*

Accordingly, the discovery of the non-covered prevalent co-location set Ω has to start from the largest size of CPCs. At the same time, hard co-locations need to be added dynamically.

For the convenience of computing *ESD* and identifying non-covered co-locations, we introduce a new concept and a Lemma.

Definition 12 (Set Participation Ratio). *The* set participation ratio SPR *(*$\{c_1, c_2, \dots, c_t\}, f$*) of the common feature f in* $\{c_1, c_2, \dots, c_t\}$ *is the fraction of instances of f that occur in* $\bigcup_{j=1}^{t}\Pi_{f_i}T(c_j)$*, that is,*

$$SPR(\{c_1,c_2,\dots,c_t\},f_i) = \frac{|\bigcup_{j=1}^{t}\Pi_{f_i}T(c_j)|}{\text{Total number of instances of } f_i}, \quad (6)$$

where $\Pi_{f_i}T(c)$ *is the set of distinct instances of* f_i *in* $T(c)$*, and* $T(c)$ *is the co-location instance of c.*

For example, consider the data set in Fig. 1a, and $c_1 = \{\text{A, C, D}\}$ and $c_2 = \{\text{B, C, D}\}$ in Fig. 1b. Since $\Pi_C T(c_1) = \{\text{C.1, C.3}\}$, $\Pi_C T(c_2) = \{\text{C.1, C.2}\}$ and C has three instances, we have $SPR(\{c_1, c_2\}, \text{C}) = 3/3 = 1$.

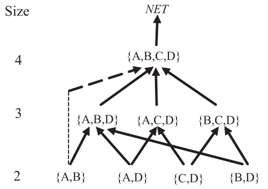


Fig. 2. The *NET* of *CPC* for the data set in Fig. 1a.

Lemma 2. *For any co-location pattern c in Ω, if its super-pattern set is* $\{c_1, c_2, \dots, c_t\}$*, then* $\hat{M} < 1-\delta$ *holds, where* $\hat{M} = \max_{f_i\in c}\{\frac{SPR(\{c_1,c_2,\dots,c_t\},f_i)}{PR(c,f_i)}\}$.

Proof. Since $c \in \Omega$, we have $ESD(c, \{c_1, c_2, \dots, c_t\}) > \delta$. So,

$$\delta < ESD(c,\{c_1,c_2,...,c_t\}) = \min_{f_i\in c}\left\{\left(1 - \frac{|\bigcup_{j=1}^{t}\Pi_{f_i}T(c_j)|}{|\Pi_{f_i}T(c)|}\right)\right\}$$

$$= 1 - \max_{f_i\in c}\left\{\frac{|\bigcup_{j=1}^{t}\Pi_{f_i}T(c_j)|}{|\Pi_{f_i}T(c)|}\right\} = 1 - \hat{M},$$

where $\hat{M} = \max_{f_i\in c}\{\frac{SPR(\{c_1,c_2,\dots,c_t\},f_i)}{PR(c,f_i)}\}$.

That is, $\hat{M} < 1-\delta$ holds. □

According to Definitions 9, 10 and the proof of Lemma 2, the condition $\hat{M} < 1-\delta$ is an iff condition with respect to whether a co-location c remains in Ω.

Discussion. (1) The number of hard co-locations in a data set is usually hard to estimate. From a large number of experiments in Section 5, we found this number to be very small. (2) Because of the *limited-transitivity* in Theorem 2, the set of non-covered prevalent co-locations Ω of a spatial data set *D* should not be unique. Finding the optimal solution is NP-hard, so the two algorithms presented in Sections 3 and 4 are aimed at obtaining a reduced result with respect to the original collection of CPCs.

3 The RRclosed Method

In this section, we present the RRclosed method which adopts a *post-mining* framework to implement the prevalent co-location redundancy reduction from *CPC* which is a collection of closed prevalent co-locations.

The RRclosed method consists of two stages. Stage 1 constructs a *net structure NET* of *CPC* according to the relationship of patterns and super-patterns in order to facilitate the next stage, and Stage 2 reduces redundant co-locations that are **δ**-covered by their super-patterns.

In Stage 1, we adopt a method beginning from the largest-size co-locations of *CPC*. When inserting a size-*l* co-location *c* into *NET*, we compute *DEC*(*c*), which is the collection of the super-patterns of *c*, by using the intersection operation of *c* with $l+1$ size co-locations in *NET*. If $DEC(c) = \phi$, $l+2$ size co-locations are considered, until reaching the root of *NET*. For example, for the spatial data set in Fig. 1a, if $M = 0.3$ then $CPC = \{\{\text{A, B, C, D}\}, \{\text{A, B, D}\}, \{\text{A, C, D}\}, \{\text{B, C, D}\}, \{\text{A, B}\}, \{\text{A, D}\}, \{\text{B, D}\}, \{\text{C, D}\}\}$. The *NET* of *CPC* is shown in Fig. 2.

In Stage 2, we check co-locations in *NET* starting from the largest size minus one. According to Lemma 2, we calculate $\hat{M}$ of checked co-location c and prune it from *NET* if $\hat{M} \ge (1-d)$. In the realization, we can sort all features in c

in *ascending order* of the PR values, and the negative condition in Lemma 2 (i.e., $\hat{M} \geq (1-d)$) may be satisfied earlier. When a co-location c is pruned from *NET*, it is necessary to revise the relationships of patterns and super-patterns affected by c, and insert the *hard co-locations* of c into *NET*. For example, in Fig. 2, if {A, B, D} is covered by its super-pattern {A, B, C, D}, its sub-pattern {A, B} will be directly connected to {A, B, C, D} (see the dotted line in Fig. 2) because {A, B, D} is a unique super-pattern of {A, B}.

The full RRclosed algorithm is summarized in Algorithm 1.

Algorithm 1. RRclosed

Input: *(1) A collection of closed prevalent co-locations CPC and their co-location instances; (2) a minimum PI threshold,* M*; (3) a coverage measure threshold,* δ.
Output: *a set of non-covered prevalent-co-locations* Ω.
Method:

```
BEGIN
   //Construct a net structure representing inclusion relationships
1)  Initiate an empty net structure NET;
2)  $l_{\max} = \text{largest\_size}(CPC)$;
3)  $l = l_{\max}$;
4)  while ( $l > 1$ or $CPC \neq \phi$ ) do
5)     for each l-size co-location c in CPC;
6)        NET.addRelations(c, DEC(c));
          //DEC(c) is the collection of the super-patterns of c
7)     $l = l - 1$;
   //Reduce the redundant co-locations
8)  $l = l_{\max} - 1$;
9)  while ( $l > 1$) do
10)    for each l-size co-location c in NET
11)    $\hat{M} = \max_{f\in c}\{\frac{SPR(DEC(c),f)}{PR(c,f)}\}$;            //see Lemma 2
12)    if $\hat{M} \geq (1-d)$ then                        //c is δ-covered
13)       NET.prune(c);
14)          if $l > 2$ then
15)             for each c′ connected only to c in NET
                   // c is the only super-pattern of c′ in NET
16)                NET.updateRelations(c′, DEC(c));
17)             NET.addHard(c);
                // the hard co-locations of c are added into NET
18)    $l = l - 1$
19) Ω ← NET //convert the structure NET into the set Ω
20) Output Ω
END
```

The computational cost of RRclosed mainly comes from the second stage, whose *coarse* computation complexity is $O(\sum_{c\in CPC}|T(c)|^r)$, where $T(c)$ is the co-location instance of c, r is the average time of scanning $T(c)$ (some $T(c)$ may be scanned many times for calculating $\hat{M}$ in Step 11, but the average scanning time r is not too big, just slightly greater than 1 time unit in general). The running time of RRclosed is principally affected by the *minimum PI threshold M* and *spatial neighbor distance threshold d*, because they control $T(c)$, the number of *CPC* and the largest size of *CPC*.

However, RRclosed needs to first compute the *CPC*. To improve the computation of the set Ω, a new method RRnull is presented in the next section.

Trans. No.	Neighbor features		Trans. No.	Neighbor instances	
1	A	B,C,D	1	A.1	B.1,C.1,D.1
2	A	B,C,D	2	A.2	B.1,B.2,C.1,D.2
3	A	B	3	A.3	B.3
4	A	C,D	4	A.4	C.3,D.4
5	B	A,C,D	5	B.1	A.1,A.2,C.1,D.1,D.2
6	B	A,C,D	6	B.2	A.2,C.1,D.2
7	B	A	7	B.3	A.3
8	B	C,D	8	B.4	C.2,D.3
9	B	D	9	B.5	D.3
10	C	A,B,D	10	C.1	A.1,A.2,B.1,B.2,D.1,D2
11	C	B,D	11	C.2	B.4,D.3
12	C	A,D	12	C.3	A.4,D.4
13	D	A,B,C	13	D.1	A.1,B.1,C.1
14	D	A,B,C	14	D.2	A.2,B.1,B.2,C.1
15	D	B,C	15	D.3	B.4,B.5,C.2
16	D	A,C	16	D.4	A.4,C.3

(a) feature neighborhood transactions (*FNT*) (b) neighborhood transactions (*NT*)

Fig. 3. The feature neighborhood transactions and neighborhood transactions of the data set in Fig. 1a.

4 The RRnull Method

In this section, we introduce a new method called RRnull which pushes the redundancy (coverage) validation into the prevalent co-location mining process.

4.1 RRnull Method

Computing the CPCs and their co-location instances is necessary for RRclosed, since the method checks coverage information based on the mined CPCs. To develop a much more efficient method, we introduce a lexicographic prefix-tree structure to store feature neighborhood transactions. We start with the generation of *non-covered* co-location candidates.

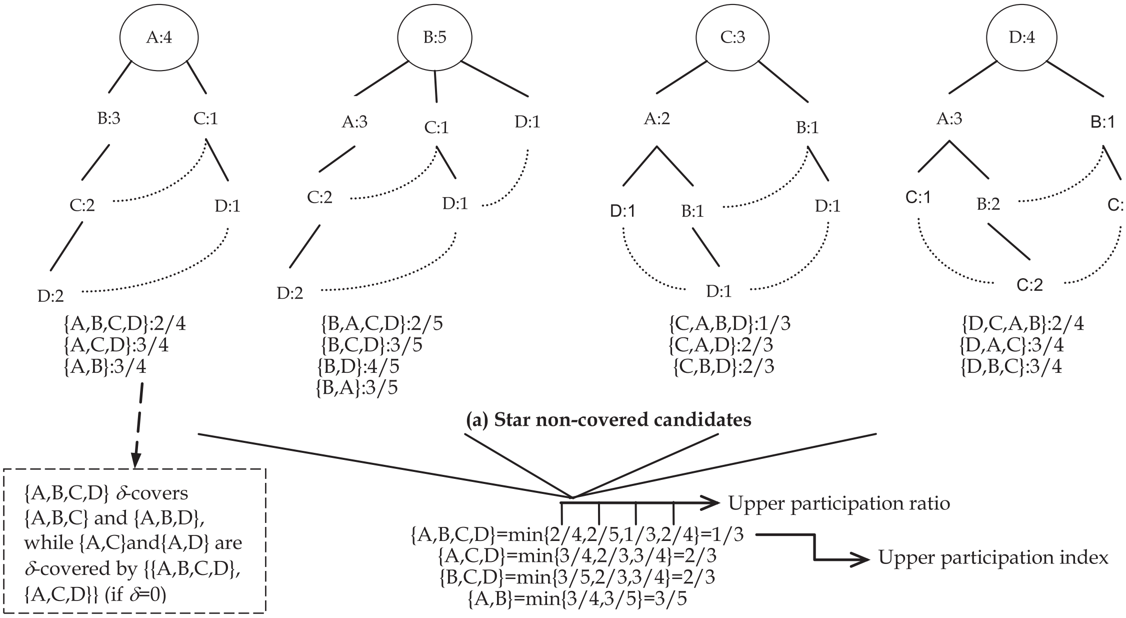
According to the related definitions, the coverage metrics are based on the co-location instances that contain the spatial neighbor relationships of instances in co-locations. So, we convert the input data to neighborhood transactions.

Definition 13 (Neighborhood transaction (NT)). *Given a spatial instance $f.i \in S$, the* neighborhood transaction *(NT) of $f.i$ is defined as a set consisting of $f.i$ and the other spatial instances having neighbor relationships with $f.i$, i.e., $NT(f,i) = \{f.i, g.j \in S \mid NR(f.i, g.j) = true \cap f \neq g\}$, where NR is a spatial neighbor relationship.*

For example, in Fig. 1a, the neighbor transaction of A.1 is {A.1, B.1 C.1, D.1}, including itself as shown in Fig. 3b. Note that each instance in the transaction has a neighbor relationship with the first instance, which is called a *reference* instance.

This data structure was first introduced in [7], [10]. It gives several advantages for *non-covered* co-location pattern mining. First, the neighborhood transactions do not lose any instances, nor lose any neighbor relationships of the original data. Second, the neighborhood transactions can be easily constructed from the neighboring instance pairs of the input data. Third, the neighborhood transactions can give the information about the upper bound value of the PI of a candidate. Finally, the feature neighborhood transactions, which are the set of distinct features in the neighborhood transactions, can be used to generate *non-covered* co-location candidates.

Definition 14 (Feature neighborhood transaction (FNT)). *The lexicographic set of distinct features in NT is called* feature neighborhood transaction.



(a) Star non-covered candidates

(b) Clique non-covered candidates

Fig. 4. Candidate generation.

The feature neighborhood transactions relative to the neighborhood transactions in Fig. 3b are shown in Fig. 3a.

The candidate generation method in [7] considers feature sets having possible clique relationships as candidates. However, here we consider feature sets having "*non-covered + possible clique relationships*" as candidates. For this, we revise the procedure proposed in [7] in three ways, as follows:

First, we generate feature sets for *star non-covered candidates* from FNTs using a lexicographic prefix-tree structure. This lexicographic prefix-tree is defined as: (1) It consists of one root labeled as a reference feature and a set of feature neighbor relationships as the children of the root, (2) Each node consists of three fields: feature-type, count and node-link, where feature-type denotes a feature that this node represents, count registers the number of neighborhood transactions represented by the portion of the path reaching this node, and node-link links to the next node in the tree carrying the same feature-type. As shown in Fig. 4a, one prefix-tree per feature is built.

Definition 15 (Star non-covered candidate (SNCC) and upper participation ratio (UPR)). *A feature set having relationships with the root node (reference feature) in a lexicographic prefix-tree is called a* star non-covered candidate *if its star participation ratio is greater than or equal to M and it has not been δ-covered by longer candidates in this prefix-tree. The star participation ratio represents the upper bound of the participation ratio of the reference features, which is the fraction of the count of reference feature in the count of neighborhood transations of all other features in a candidate. It is called the* upper participation ratio.

SNCCs are generated using the following method.

1) Each branch in a lexicographic prefix-tree forms a SNCC if its *UPR* is greater than or equal to *M*.
2) The sub-sets of the branches, which contain the root node, form SNCCs if they are not **δ**-*covered* by generated longer size candidates in the prefix-tree.

For example, in the prefix-tree of feature A, we can generate two candidates $\{A, B, C, D\} : 2/4$ and $\{A, C, D\} : 3/4$ from the two branches {A, B, C, D} and {A, C, D}, and if $\delta = 0$ then the sub-set {A, B} of branch {A, B, C, D} is also a candidate $\{A, B\} : 3/4$ since $\frac{UPR(\{A,B,C,D\},A)}{UPR(\{A,B\},A)} = \frac{2}{3} < 1 - \delta$ (see Lemma 2). The star neighborhood information of the remaining sub-sets, which contain the root node, has been **δ**-covered by generated candidates. For example, the star neighborhood information of the sub-sets {A, B, C}, {A, B, D} in branch {A, B, C, D} is the same as that of {A, B, C, D}, while the star neighborhood information of the sub-sets {A, C}, {A, D} are covered by the candidates {A, B, C, D} and {A, C, D}. To form sub-sets in a branch we conduct breadth-first enumerations in the branch set except for the root node.

Second, the SNCCs are combined for filtering the *clique non-covered candidates* (CNCCs).

Definition 16 (Clique non-covered candidate and upper participation index (UPI)). *A size-k candidate combining from k size-k SNCCs in the prefix-trees is called a size-k* clique non-covered candidate*. The minimum value of k UPRs is called the* upper participation index (UPI).

For example, in Fig. 4b, to be {A, B, C, D}, which is a clique non-covered candidate, four SNCCs {A, B, C, D}, {B, A, C, D}, {C, A, B, D} and {D, A, B, C} are needed. The sub-set {B, D} is a star non-covered candidate in prefix-tree B, but {D, B} is not a star non-covered candidate in prefix-tree D. After the combining step, {B, D} is pruned.

Once a candidate is pruned, its covered sub-sets need to be generated. For example, if $M = 0.4$, the generated candidate set in the prefix-tree C would be {{C, A, D}: 2/3, {C, B, D}: 2/3, {C, D}: 3/3}. The clique non-covered candidate {A, B, C, D} cannot be formed in the combining step, as the covered sub-sets {A, B, C}: 2/4 and {A, B, D}: 2/4 of the {A, B, C, D} in the prefix-tree A need to be generated. Prefix-tree B and D are treated similarly. The process of filtering CNCCs is conducted by dynamically changing SNCCs.

Third, the true PIs of candidates are computed starting from the largest size candidates. The *candidate co-location instances* of candidates are gathered by scanning NTs in Fig. 3b, even though they are not the true co-location instances. True co-location instances can be filtered from the candidate instances by examining clique relationships among other instances of the candidate co-location instance, except for the first instance. For example, in Fig. 3b, {A2, B2, C1, D2} is a true co-location instance of candidate {A, B, C, D}, but {A2, B1, C1, D2} is not.

For a candidate c, if PI(c) = UPI(c) then c must be a non-covered co-location. Otherwise, we first need to generate the covered sub-sets of c since these sub-sets have been pruned assuming the condition $\mathrm{UPI}(c) = \mathrm{PI}(c)$; next, if $\mathrm{PI}(c) < M$ then c can be pruned out; else, we need to check whether c is δ-covered by its super-patterns or not.

Note that the UPI values of size 2 co-locations are the true PI values.

4.2 The Algorithm

Algorithm 2 shows the pseudo code of the RRnull method.

Algorithm 2 contains three stages. The first one is to preprocess and generate *SNCC*, the second one is to combine *SNCC* as *CNCC*, and the third one is to calculate true PI values of candidates. In the second and the third stages, once a candidate can be pruned out, its covered sub-patterns need to be generated.

In Stage 1, we first find all neighboring instance pairs for a given input spatial data set. The neighborhood transactions are generated by grouping the neighboring instances per instance. Then, a prefix-tree per feature is built with the lexicographic neighborhood transactions. The set of SNCCs is generated based on the prefix-trees.

In Stage 2, the set of CNCCs is filtered by combining the SNCCs. The UPI of each candidate in *CNCC* is computed. In this phase, if candidates in *SNCC* cannot be combined, their smaller size covered sub-patterns are generated. The combining step starts from the largest size of *SNCC*.

The third stage is to calculate the true PIs of candidates in *CNCC*. First, the star instances of a candidate are found by scanning neighborhood transactions. Then, the clique instances can be filtered from the star instances by examining a clique relationship among other instances, except for the first instance of the star instance. Next, the true PIs can be calculated based on the clique instances of candidates. For a candidate c, if PI(c) = UPI(c) then the candidate can be moved from *CNCC* to the non-covered prevalent co-location set Ω. However, if PI(c) ≠ UPI(c), we have to do some further work as shown in Algorithm 2.

4.3 The Analysis and Comparison

4.3.1 The Correctness Analysis

Although Algorithm 2 seems simple, it can completely eliminate redundancy and get the correct non-covered prevalent co-location set Ω, i.e., Algorithm 2 works. The reasons are as follows.

1. A co-location instance must be a star neighborhood instance, and it corresponds to a feature neighborhood transaction in Algorithm 2.
2. According to the concept of δ-covered, for a co-location c, if the all instances of one feature in $T(c)$ are δ-covered by its super-pattern, c is δ-covered. So we can introduce the SNCC and CNCC in Algorithm 2.
3. For the sub-patterns δ-covered by a set of super-patterns, because the generation of *SNCC* is based on the prefix-tree of the reference feature, if the co-location instances of the reference feature are δ-covered by its super-patterns then it must not appear in the *SNCC*. That is why we have Step 22 in Algorithm 2.
4. When PI(c) ≠ UPI(c), Step 23 generates $(l-1)$-size sub-patterns covered by c, and Step 26 checks whether c is δ-covered by its super-patterns or not.

Algorithm 2. RRnull

Input: *(1) A feature set, $F = \{f_1, f_2, \ldots, f_n\}$; (2) A spatial instance set, S; (3) a spatial neighbor distance threshold, d; (4) a minimum PI threshold, M; (5) a coverage measure threshold, δ.*
Output: *the non-covered prevalent co-location set Ω.*
Method:

```
BEGIN
 // Preprocess and generate star non-covered candidates
 1)  $NP = \mathrm{find\_neighbor\_pairs}(S, d)$;
 2)  $(NT, FNT) = \mathrm{gen\_neighbor\_transactions}(NP)$;
 3)  for $i = 1$ to $n$
 4)    $Tree_i = \mathrm{build\_prefix\text{-}tree}(f_i, FNT)$;
 5)    $SNCC = \mathrm{gen\_candi\_and\_cal\_upr}\,(tree_1, .., tree_n, M)$;
              // Generating star non-covered candidates with UPR
     // Combining step
 6)  CNCC←size 2 candidates in SNCC;
              //CNCC is the set of clique non-covered candidates
 7)  $l$ = largest size of SNCC;
 8)  while ($l > 2$ and $SNCC \neq \phi$) do
 9)    for l-size candidates in SNCC;
10)      CNCC←combine_and_cal_upi (SNCC);
11)      if l-size candidate c in SNCC cannot be combined
12)      then $SNCC \leftarrow \mathrm{gen\_covered\_}(l-1)\text{-sub-sets}(c,\ SNCC)$;
13)    $l = l - 1$;
   // calculate true PIs of candidates
14)  Ω←size 2 candidates in CNCC;
15)  $l$ = largest size of CNCC;
16)  while ($l > 2$ and $CNCC \neq \phi$) do
17)    for each l-size candidate c in CNCC;
18)      $SI_c = \mathrm{find\_star\_instances}(c,\ NT)$;
19)      $CI_c = \mathrm{filter\_clique\_instances}(SI_c,\ NT)$;
20)      $PI(c) = \mathrm{calculate\_true\_pi}(CI_c)$;
21)      if $PI(c) = UPI(c)$
22)      then move c from CNCC to Ω
23)      else $CNCC \leftarrow \mathrm{gen\_covered\_}(l-1)\text{-sub-sets}(c,\ CNCC)$;
24)        if $PI(c) < M$
25)          then prune_candidate(CNCC)
26)          else if $c$ is not covered                    //by Lemma 2
27)             then $\Omega \leftarrow c$
28)    $l = l - 1$;
29)  Output Ω
END
```

4.3.2 The Time Complexity Analysis

Algorithm 2 has three main parts: generating *SNCC* (Steps 1-5), filtering *CNCC* (Steps 6-13) and calculating the true PIs

(Steps 14-29). The complexity analysis of the three parts is shown as follows.

Generating SNCC. In order to generate *SNCC*, RRNull first generates *NT* and *FNT* (Steps 1-2), and then builds the prefix-trees for each feature (Steps 3-4). Finally, *SNCC* is generated from the prefix-trees (Step 5).

- Generating *NT* and *FNT*: A grid-based method is used to calculate the neighborhood relationships. The whole space is divided into grids with area d^*d (d is the spatial neighbor distance threshold), such that every instance in a certain grid g is only compared with the instances located in 4 grids around g (east, southeast, south and southwest). *FNT* is generated during the generation of *NT*, and a neighborhood relationship will be added to both *NT* and *FNT*, so that the computational complexity of generating *NT* and *FNT* is about: $O(n^2 * \frac{d^2}{A})(\frac{d^2}{A} < < 1)$, where d is the distance threshold, n is the number of instances and A is the area of the whole space.
- Building prefix-trees: If the number of features is m, m prefix-trees are built based on *FNT*. We note that the upper value of the number of *FNT* is n (n is the number of instances). Thus, to build m prefix-trees based on the *FNT*, the computational complexity is about: $O(size_{avg}(FNT)^*|FNT|)$, where $size_{avg}$ (FNT) is the average size of the transactions in *FNT*, and $|FNT| \approx n$.
- Generating *SNCC*: SNCCs are always found from the leaf nodes in RRnull, for example, for the first prefix-tree in Fig. 4a, there are 2 leaf nodes (D: 2 and D: 1), for the leaf node D: 2, we generate a SNCC $\{\mathrm{A, B, C, D}\} : 2/4$, remove D: 2 and the count value of the leaf node's ancestors minus the leaf node's count value of 2. Then every node with a count value 0 is removed from the prefix-tree (covered). Thus, a SNCC {A, B, C, D}: 2/4 is generated and the node C: 2 is removed. The process will continue until the prefix-tree is empty. The method performs well and the computational complexity in the worst case where only the leaf node is removed when getting a SNCC is: $O((size_{\mathrm{avg}}(FNT))^{2} * n')$, where n' is the number of branches of the all prefix-trees ($n' < n$).

In summary, the computational complexity of generating *SNCC* is about:

$$O\left(n^2 * \frac{d^2}{A}\right) + O((size_{avg}(FNT))^2 * n).$$

Filtering CNCC. In the process of generating *SNCC*, the participation ratio of each SNCC is stored in a hash set h wherein the key of h is the SNCC, and the value is the set of participation indexes whose first value is the minimum value of the whole set. For example, if {A, B, C, D} is generated from the first prefix-tree in Fig. 4a, its participation ratio will be added to h as [key: {A, B, C, D}, value: $\{2/4, 2/4\}$]. If the participation ratio of {A, B, C, D} calculated from the second prefix-tree in Fig. 4a is $2/5$, it will be added to h as [key: {A, B, C, D}, value: $\{2/5, 2/4, 2/5\}$]. After the process of generating *SNCC*, h is used to filter *CNCC*. If the number of participation ratios of one candidate c is less than the size of c, c will be pruned. Otherwise, the UPI of c can be quickly obtained from the first element of the value attribute of c in h, and then c is regarded as a CNCC. The main cost of this process is the traverse of h, so the computational complexity of this part is about: $O(|SNCC|)$.

Calculating the true PIs. To calculate the true PI of a co-location c, the co-location instance $T(c)$ of c is needed. For each candidate c in *CNCC*, based on *NT*, a joinless approach is used to generate $T(c)$. The computational complexity is about: $O(|CNCC|*(size_{\mathrm{avg}}(CNCC))^2 * l_{avg}(CNCC|_{\mathrm{instance}}))$, where $size_{avg}(CNCC)$ is the average size of the candidates in *CNCC* (*t*he average size of candidates in *CNCC* is close to the average size of transactions in *FNT*), and l_{avg} $(CNCC|_{\mathrm{instance}})$ is the average count of the co-location instances of candidates in *CNCC*, and that in general $(l_{avg}(CNCC|_{\mathrm{instance}}) > n)$.

Combining the above analysis of the three parts in RRNull, the final computational complexity of RRNull is about: $O(n^2 * \frac{d^2}{A}) + O((size_{avg}(FNT))^2 * n) + O(|SNCC|) + O(|CNCC| * (size_{avg}(CNCC))^2 * l_{avg}(CNCC|_{\mathrm{instance}})) \approx O(|CNCC|* (size_{avg}(CNCC))^2 * l_{avg}(CNCC|_{\mathrm{instance}}))$.

Obviously, the computational complexity of RRnull is dominated by the third part.

4.3.3 Comparitive Analysis

The running time of RRnull is much faster than that of CPC mining. The comparative analysis is as follows:

1. If a co-location is covered by its super-patterns, it might be eliminated in the generating star non-covered candidates' stage, in the combining stage, or in the stage of calculating true PIs. If the eliminated patterns were equally distributed in the three stages, about $2/3$ of the co-location instances of the covered patterns are no longer calculated. Indeed, more covered co-locations may be eliminated in advance. For the data set in Fig. 1a, all covered co-locations have been eliminated before the combining step.
2. The *non-covered* condition is stronger than the *closed* condition, so the candidate set generated in Algorithm 2 must be smaller than that in CPC mining.
3. The *size* of the covered co-locations is smaller than that of their super-patterns. We note that the smaller the size of co-locations, the larger the number of co-location instances.
4. When there are plenty of star co-location instances that are not true co-location instances, the work of generating and checking their sub-patterns is time consuming. However, the corresponding problem also appears in CPC mining. Therefore, in this case the running time of Algorithm 2 is still faster than that of the CPC mining.

5 Experimental Results

In this section, we design sets of experiments to test the performance of the proposed algorithms. We use three real datasets and a series of synthetic datasets in our experiments. The proposed algorithms are implemented in Visual C#. All of our experiments are performed on an Intel PC running Windows 7 with Intel Core i5 3337U @ 1.80 GHz CPU and 2 GB-memory.

TABLE 1
A Summary of the Three Real Data Sets

Name	N. of features	N. of instances	(Max, Min)	The distribution area of spatial instances (m^2)
Real-1	32	335	(63, 3)	80000 × 130000
Real-2	20	377834	(60000, 347)	50000 × 80000
Real-3	15	501046	(55646, 8706)	110000 × 160000

(Max, Min): are respectively the maximum number and the minimum number of the feature's instances in the data sets.

5.1 On the Three Real Data Sets

This section examines the performance of the proposed algorithms on the three real data sets. The first tests the redundancy reduction power, and the second measures the computational performance of the proposed algorithms. A summary of the selected three real data sets is presented in Table 1. Real-1 is from the rare plant data of the Three Parallel Rivers of Yunnan Protected Areas whose instances form a zonal distribution as shown in Fig. 5a. Real-2 is a spatial distribution data set of urban elements whose instances' distribution is both even and dense as shown in Fig. 5b. Real-3 is a vegetation distribution data set of the Three Parallel Rivers of Yunnan Protected Areas, which has the fewest features but the most instances, its instance distribution being various clusters as shown in Fig. 5c.

5.1.1 The Power of Redundancy Reduction

For each real data set, we vary the values of parameters M (the minimum PI thresholds), d (the spatial neighbor distance thresholds) and δ (the coverage measure thresholds) respectively to verify the redundancy reduction power of our method with respect to the original closed prevalent co-locations $(|S_{closed}| - |\Omega|) \,/\, |S_{closed}|$, where S_{closed} is the set of CPCs and Ω is the non-covered prevalent co-location set produced by Algorithm 1 or Algorithm 2. The experimental results are shown in Figs. 6a to 6f. We have the following observations: First, RRclosed and RRnull generate the same results on all real data sets; second, on all three real data sets, the redundancy reduction power is between 10 and 85 percent, the mean value being about 56 percent. On the Real-2 data set, the effect of redundancy reduction is the best, and its mean redundancy reduction power reaches 65 percent. This is because our redundancy reduction method is based on utilizing distributed information about co-location instances, and Real-2 is an evenly distributed data set; third, the redundancy reduction power becomes large when M is low or d is large. That is expected because there are more CPCs mined under lower M or larger d; fourth, as we expected, the redundancy reduction power increases when the value of δ increases. But the difference is not very big. This result illustrates that the value of δ is not the main

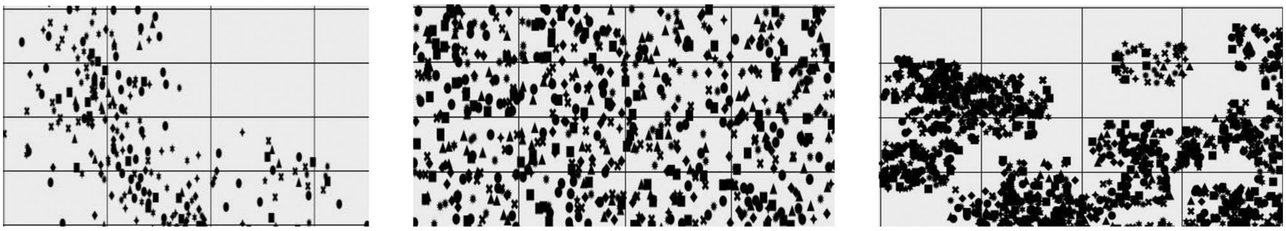

(a) part of distribution of Real-1 (b) part of distribution of Real-2 (c) part of distribution of Real-3

Fig. 5. Spatial distribution of the three real data sets.

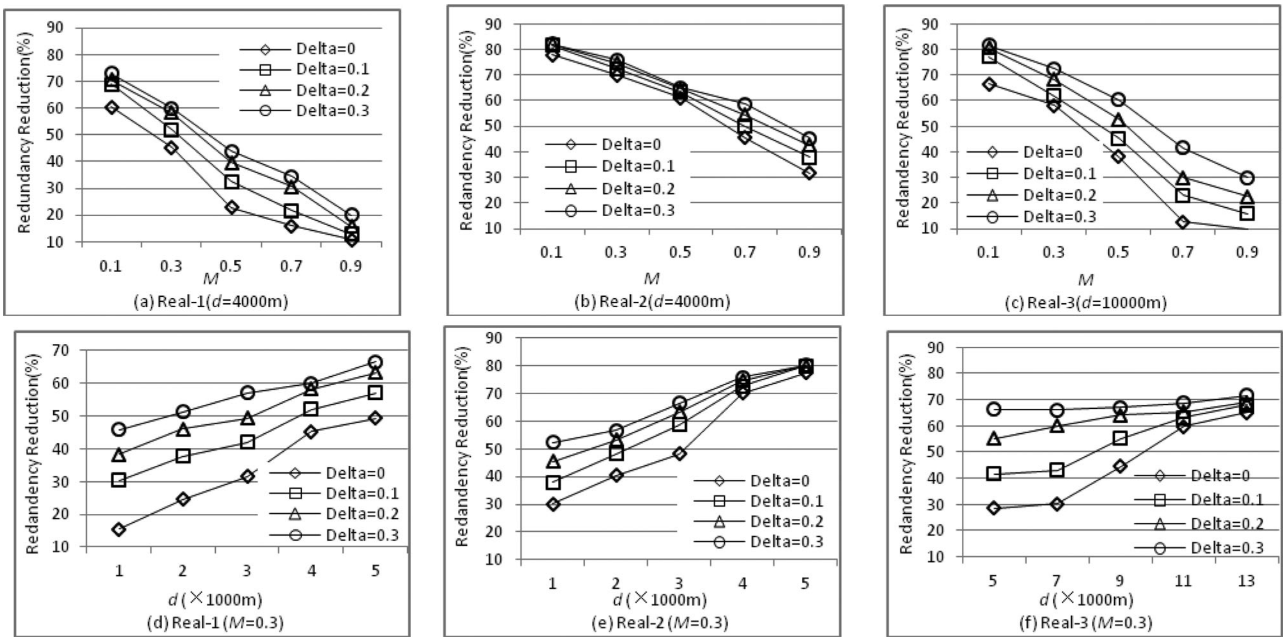


Fig. 6. The Power of redundancy reduction over the three real data sets.

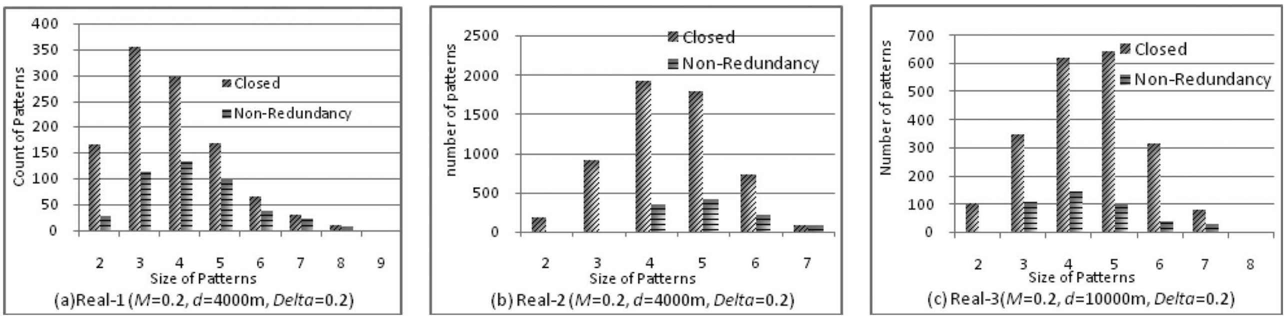


Fig. 7. Comparison of the redundancy reduction power over different sizes of patterns.

factor that affects the redundancy reduction power. In the experiments, we set δ as 0, 0.1, 0.2 and 0.3 respectively.

The comparisons of the redundancy reduction power over different sizes of patterns are shown in Fig. 7a to Fig. 7c. As can be seen, the longest CPCs are kept because no patterns can contain them, while for each size from maxlen-1, the number of kept patterns is less, or even much less, than the number of CPCs. Usually, patterns are reduced mostly in the middle sizes, e.g., patterns with size 4 and size 5 in Fig. 7b are reduced the most.

5.1.2 *The Running Time*

The corresponding running time of the three methods, RRclosed, RRnull and Closed which is a CPC mining algorithm presented in [7], are shown from Figs. 8a to 8f. The running time of RRclosed includes the Closed procedure, which we have optimized.

The results show that RRclosed is much slower than RRnull, especially when M is low and d large. Comparing RRnull and Closed, we observe that RRnull runs much better than Closed especially on dense data sets (a larger d makes a denser data set). This is because RRnull examines the δ-covered co-locations, while Closed examines the co-locations' closeness, and the former condition is stronger than the latter's. We further observe that RRnull runs two times faster than Closed in Real-2 when $M = 0.3$ in Fig. 8b, or when $d = 3000\,\mathrm{m}$ and $4000\,\mathrm{m}$ in Fig. 8e, and three times faster in Real-2 when $M = 0.1$ in Fig. 8b, or when $d = 5000$m in Fig. 8e. Because RRnull avoids identifying many candidates, it also saves much space. Finally, from the running times of RRclosed and Closed which are almost identical, we can verify that Algorithm 1 is efficient for redundancy reduction.

5.2 On the Synthetic Data Sets

This section examines the scalability of the RRnull and the RRclosed with the varying numbers of spatial instances, numbers of spatial features, neighbor distance thresholds, prevalence thresholds, and coverage measure thresholds.

Synthetic data sets were generated using a spatial data generator similar to [6], [10]. Table 2 describes the parameters used for the data generation in the related experiments. First, the distribution area of spatial instances was determined with D^*D. The whole area was divided into grids with d^*d where d is the spatial neighbor distance threshold. Then, we generated P initial core patterns whose average size was Q. The feature types of each core pattern were

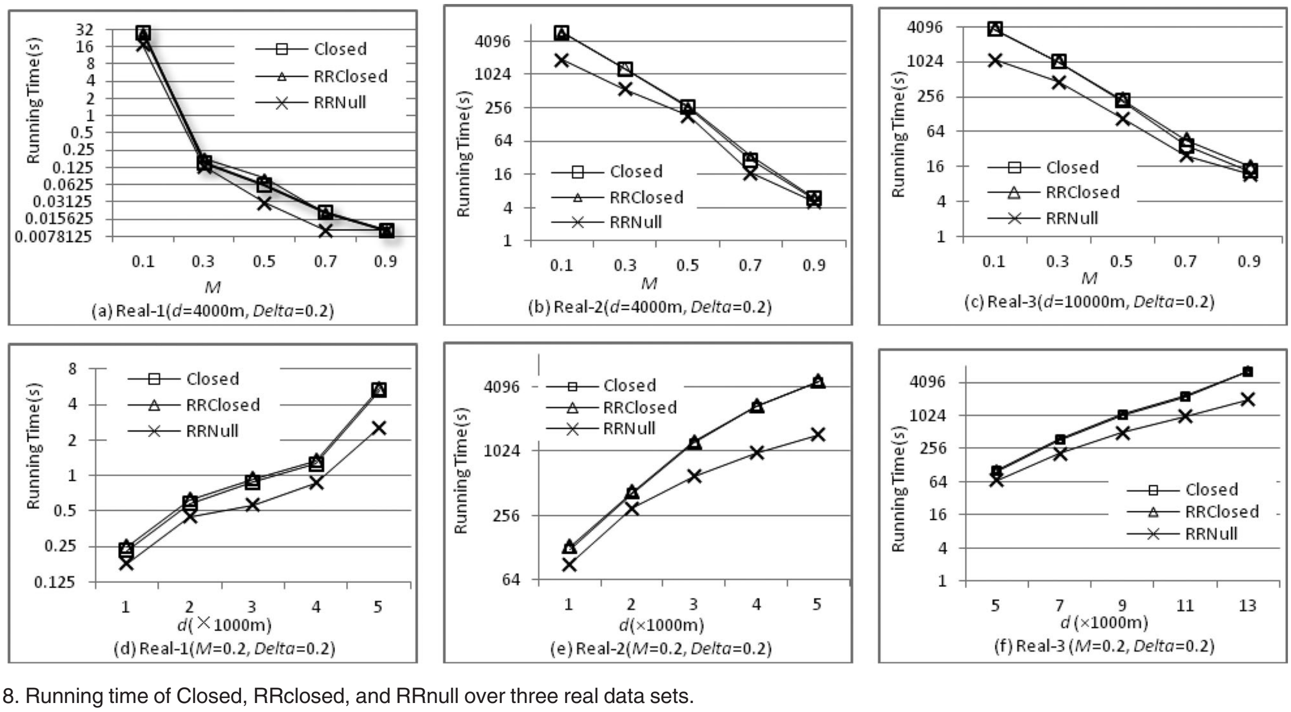


Fig. 8. Running time of Closed, RRclosed, and RRnull over three real data sets.

TABLE 2
Parameters and Their Values in Experiments

Parameter	Definition	Experiment Figures				
		F.9a	F.9b	F.9c	F.9d	F.9e
D	Spatial area ($D\times D$)	10000				
P	Number of ♀co-locations	20				
Q	Average size of ♀co-locations	5				
I	Average number of ♀co-location instances	$S/100$	2000			
S	Number of instances	*	200000			
F	Number of features	20	*	20		
d	Spatial neighbor distance threshold	1000		*	1000	
M	Minimum PI threshold	0.2			*	0.2
$\boldsymbol{\delta}$	Measure of coverage	0.2				*

♀: initial core co-location, *: variable values.

randomly chosen from F features. An average of I instances per core pattern were generated. The total number of instances was around S. Finally, the co-location instances for each initial pattern were distributed. To locate a co-location instance, we first randomly chose a grid, and all points of the instance were then randomly located within the chosen grid.

As shown in Figs. 9a to 9e, the RRnull algorithm shows scalability to large dense sets, at lower M, larger d and lower δ, and it performs better than RRclosed in all the experiments. This is because RRclosed considers feature sets having "*possible clique relationships*" as candidates, while RRnull takes "*non-covered + possible clique relationships*" as candidates. When the number of spatial instances is 500,000 as in Fig. 9a, RRnull runs almost *five* times faster than RRclosed. We compare the number of CPCs, the number of RRnull candidates and the number of non-covered co-locations over different sizes in Fig. 10. From the figure, we can see that the number of RRnull candidates is much lower than that of CPCs. Also, we can see that with the size of candidates growing, the number of RRnull candidates gets closer and closer to the final result. Clearly, checking a longer pattern costs much more time than checking a shorter one.

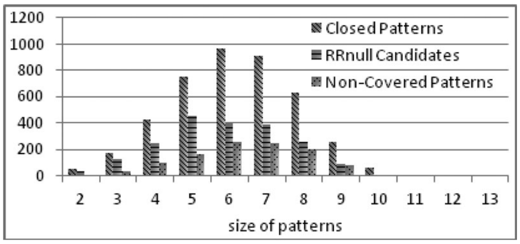


Fig. 10. Further analysis of the 500,000 instances in Fig. 9a.

In addition, we notice that the trend in Fig. 9b does not increase progressively as the number of features grows. This is because we have fixed the total number of instances, so the number of instances for each feature reduces successively when the number of features grows. When the number of features exceeds 40, the number of co-location instances and the number of candidates decreases sharply.

We have further investigated the reason for speed increase of the RRnull algorithm. Table 3 compares the numbers of the generated candidates by RRnull and Closed at several special places found in Fig. 9. We can see that RRnull identifies about 50 percent of the false candidates before calculating their true co-location instances.

6 Related Work

In Section 1, we have discussed the connection of our work with previous prevalent co-location mining, with maximal prevalent co-location mining and with closed prevalent co-location mining. A closely related work is the top-k closed co-location mining problem studied in [7], where the

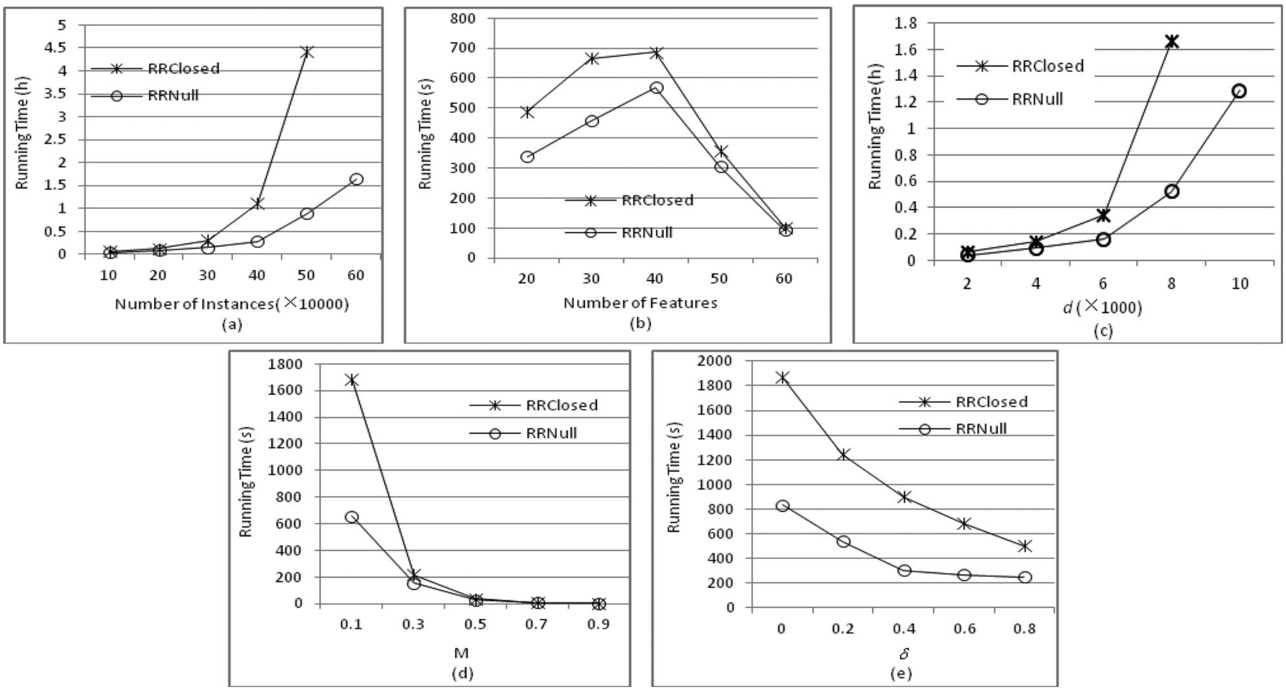


Fig. 9. Scalability analysis on the synthetic data sets.

TABLE 3
Comparison of the Number of Candidates Generated by RRnull and Closed

N. of candidates	$\mid S \mid = 500000$ in Fig. 9a	$\mid F \mid = 30$ in Fig. 9b	$d = 8000$ in Fig. 9c	$M = 0.1$ in Fig. 9d	$\boldsymbol{\delta} = 0$ in Fig. 9e
RRnull	2039	3201	2573	1236	1319
Closed	5285	5396	5538	2651	2944

criterion of the top-k co-locations is to provide best prevalence estimation of those co-locations that are closed and not selected. This paper's approcah is more sophisticated since the co-locations reduced by our method are δ-covered by their super-patterns with respect to their distributed information about the co-location instances.

The motivation for our work is the same as that in [11], [12], [13], [14], [15] but now our redundant co-location concept is based on spatial distribution information of co-location instances, dissimilar to the techniques used in transaction data sets. Further, our proposed new algorithms can reduce the size of the original collection of closed co-locations by about 50 percent.

Our work shares some common interests with the work in [16], [17], [18], [19], [20] because they all take spatial distribution of co-location instances into consideration. However, our work is totally different from these works, in which the spatial instance distribution information is integrated into prevalence metrics or neighborhood distance measures. We did not change the classical prevalence metrics expressed in [1], [2], [3], [4], [5], [6], [7], [8], [9], [10]. What we have studied is the relationship between co-location patterns by introducing the $\boldsymbol{\delta}$-covered concept based on the spatial distribution of co-location instances. Our purpose is to discover more succinct compression of the non-redundant co-location sets.

Recently domain-driven pattern mining has been attracting more researchers. Many efforts [21], [22], [23], [24], [25], [26] have been devoted into prevalent co-location pattern mining, which have extended the prevalent co-location mining problem. We expect that the study of domain-driven prevalent co-location redundancy reduction will also prove to be significant.

7 Conclusion

We have considered the redundancy reduction problem of the spatial prevalent co-locations by applying distribution information from co-location instances. It is worth mentioning that the proposed method not only solves the redundancy reduction problem but also provides high efficiency.

There are several interesting directions that we are considering for future work: (1) Compression of the spatial prevalent co-locations (to get fewer co-locations but more usability, i.e., a set of representative co-locations); (2) Ordering of the spatial prevalent co-locations; and (3) Reducing the redundancy of prevalent co-locations found in incrementally updated data.

Acknowledgments

This work was supported in part by grants (No. 61472346, No. 61662086) from the National Natural Science Foundation of China, in part by a grant (No. 2016FA026, No. 2015FB114) from the Science Foundation of Yunnan Province, and in part by the Project of Innovation Team of Yunnan University.

References

[1] F. Verhein and G. Al-naymat, "Fast mining of complex spatial co-location patterns using GLIMIT," in *Proc. 7th IEEE Int. Conf. Data Min. Workshops*, 2007, pp. 679–684.

[2] J. Li, et al., "On discovering co-location patterns in datasets: A case study of pollutants and child cancers," *Geoinformatica*, vol. 20, pp. 651–692, 2016, doi: 10.1007/s10707-016-0254-1.

[3] W. Yu, "Spatial co-location pattern mining for location-based services in road networks," *Expert Syst. Appl.*, vol. 46, pp. 324–335, 2016.

[4] M. Akbari, F. Samadzadegan, and R. Weibel, "A generic regional spatio-temporal co-occurrence pattern mining model: A case study for air pollution," *J. Geograph. Syst.* vol. 17, no. 6, pp. 249–274, 2015.

[5] X. Yao, L. Chen, L. Peng, and T. Chi, "A co-location pattern-mining algorithm with a density-weighted distance thresholding consideration," *Inf. Sci.*, vol. 396, no. 2, pp. 144–161, 2017.

[6] Y. Huang, S. Shekhar, and H. Xiong, "Discovering co-location patterns from spatial data sets: A general approach," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 12, pp. 1472–1485, Dec. 2004.

[7] J. S. Yoo and M. Bow, "Mining top-k closed co-location patterns," in *Proc. IEEE Int. Conf. Spatial Data Min. Geographical Knowl. Serv.*, 2011, pp. 100–105.

[8] L. Wang, L. Zhou, J. Lu, and J. Yip, "An order-clique-based approach for mining maximal co-locations," *Inf. Sci.*, vol. 179, no. 19, pp. 3370–3382, May 2009.

[9] J. S. Yoo and M. Bow, "Mining maximal co-located event sets," in *Proc. Pac.-Asia In. Conf. Knowl. Discovery Data Min.*, 2011, pp. 351–362.

[10] J. S. Yoo and S. Shekhar, "A join less approach for mining spatial co-location patterns," *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 10, pp. 1323–1337, Oct. 2006.

[11] D. Xin, J. Han, X. Yan, and H. Cheng, "Mining compressed frequent-pattern sets," in *Proc. Int. Conf. Very Large Data Bases*, 2005, pp. 709–720

[12] X. Yan, H. Cheng, J. Han, and D. Xin, "Summarizing itemset patterns: A profile-based approach," in *Proc. 11th ACM SIGKDD Int. Conf. Knowl. Discovery Data Min.*, 2005, pp. 314–323.

[13] D. Xin, X. Shen, Q. Mei, and J. Han, "Discovering interesting patterns through user's interactive feedback," in *Proc. 12th ACM SIGKDD Int. Conf. Knowl. Discovery Data Min.*, pp. 773–778, 2006.

[14] D. Xin, H. Cheng, X. Yan, and J. Han, "Extracting redundancy-aware top-k patterns," in *Proc. 12th ACM SIGKDD Int. Conf. Knowl. Discovery Data Min.*, 2006, pp. 444–453.

[15] T. Mielikäinen and H. Mannila, "The pattern ordering problem," in *Proc. Eur. Conf. Principles Data Min. Knowl. Discovery*, 2003, pp. 327–333.

[16] C. Sengstock, M. Gertz, and T. V. Canh, "Spatial interestingness measures for co-location pattern mining," in *Proc. 12th IEEE Int. Conf. Data Min. Workshops*, 2012, pp. 821–826.

[17] F. Qian, K. Chiew, Q. He, and H. Huang, "Mining regional co-location patterns with kNNG," *J. Intell. Inf. Syst.*, vol. 42, no. 3, pp. 485–505, Mar. 2014.

[18] P. Mohan, S. Shekhar, J. A. Shine, J. P. Rogers, Z. Jiang, and N. Wayant, "A neighborhood graph based approach to regional co-location pattern discovery: A summary of results," in *Proc. 20th ACM SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst.*, 2011, pp. 122–131.

[19] M. Celik, J. M. Kang, and S. Shekhar, "Zonal co-location pattern discovery with dynamic parameters," in *Proc. 7th IEEE Int. Conf. Data Min.*, 2007, pp. 433–438.

[20] S. Barua and J. Sander, "Mining statistically significant co-location and segregation patterns," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 5, pp. 1185–1199, May 2014.

[21] L. Wang, P. Wu, and H. Chen, "Finding probabilistic prevalent co-locations in spatially uncertain data sets," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 4, pp. 790–804, Apr. 2013.

[22] L. Wang, J. Han, H. Chen, and J. Lu, "Top-k probabilistic prevalent co-location mining in spatially uncertain data sets," *Frontiers Comput. Sci.*, vol. 10, no. 3, pp. 488–503, Sep. 2015.

[23] L. Wang, W. Jiang, H. Chen, and Y. Fang, "Efficiently mining high utility co-location patterns from spatial data sets with instance-specific utilities," in *Proc. Int. Conf. Database Syst. Adv. Appl.*, 2017, pp. 458–474.

[24] Z. Ouyang, L. Wang, and P. Wu, "Spatial co-location pattern discovery from fuzzy objects," *Int. J. Artif. Intell. Tools*, vol. 26, 2017, Art. no. 1750003, doi: 10.1142/S0218213017500038

[25] J. Lu, et al., "Mining competitive pairs hidden in co-location patterns from dynamic spatial databases," in *Proc. Pac.-Asia Conf. Knowl. Discovery Data Min.*, 2017, pp. 467–480.

[26] X. Wang and L. Wang, "Incremental mining of high utility co-locations from spatial database," in *Proc. BigComp*, 2017, pp. 215–222.

Lizhen Wang received the BS and MSc degrees in computational mathematics from Yunnan University, in 1983 and 1988, respectively, and the PhD degree in computer science from the University of Hundersfield, in 2008. She is a professor in the Department of Computer Science and Engineering, Yunnan University. Her research interests include data mining, data warehouses, and computer algorithms. She is a member of the IEEE.

Xuguang Bao received the BS degree in software engineering form Wuhan University of Technology, in 2008, and the MSc degree in computer science from Yunnan University, in 2015. He has been working toward the PhD degree in computer science at Yunnan University, since 2015. His main research interest is spatial data mining.

Lihua Zhou received the BS and MSc degrees in electronics and information system from Yunnan University, in 1989 and 1992, respectively, and the PhD degree in communication and information system from Yunnan University, in 2010. She is a professor in the Department of Computer Science and Engineering, Yunnan University. Her main research interests include data mining and social network analysis.