Article

Maximal Instance Algorithm for Fast Mining of Spatial Co-Location Patterns

Guoqing Zhou ¹, Qi Li ^{2,*} and Guangming Deng ²¹ Guangxi Key Laboratory of Spatial Information and Geomatics, Guilin University of Technology, Guilin 541004, China; gzhou@glut.edu.cn² College of Science, Guilin University of Technology, Guilin 541004, China; dgm@glut.edu.cn

* Correspondence: lq751277@163.com; Tel.: +86-130-2659-9733

Abstract: The explosive growth of spatial data and the widespread use of spatial databases emphasize the need for spatial data mining. The subsets of features frequently located together in a geographic space are called spatial co-location patterns. It is difficult to discover co-location patterns because of the huge amount of data brought by the instances of spatial features. A large fraction of the computation time is devoted to generating row instances and candidate co-location patterns. This paper makes three main contributions for mining co-location patterns. First, the definition of maximal instances is given and a row instance (RI)-tree is constructed to find maximal instances from a spatial data set. Second, a fast method for generating all row instances and candidate co-locations is proposed and the feasibility of this method is proved. Third, a maximal instance algorithm with no join operations for mining co-location patterns is proposed. Finally, experimental evaluations using synthetic data sets and a real data set show that maximal instance algorithm is feasible and has better performance.

Keywords: spatial data mining; co-location pattern; co-location pattern mining; maximal clique

Citation: Zhou, G.; Li, Q.; Deng, G. Maximal Instance Algorithm for Fast Mining of Spatial Co-Location Patterns. *Remote Sens.* **2021**, *13*, 960. <https://doi.org/10.3390/rs13050960>

Academic Editor: Jaime Zabazla

Received: 6 February 2021

Accepted: 26 February 2021

Published: 4 March 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Spatial co-location mining is to mine the positive relationship of different spatial features in spatial data. It is somewhat difficult to find co-location instances since the instances of spatial features are embedded in a continuous space and share neighbor relationships. Association rule mining algorithms cannot be directly applied to co-location pattern mining since there are no pre-defined transactions in spatial data sets [1]. Thus, the following three basic approaches have been proposed for mining co-locations: Huang, et al. first proposed a join-based approach [1,2]. This approach mines co-locations by using the concept of proximity neighborhood and identifies co-location instances by joining table instances, ensuring the correctness and completeness of this approach. Then, the partial-join approach [3] and the join-less approach [4,5] were proposed for fast mining of co-location patterns. The partial-join approach first transactionizes continuous spatial data (builds cliques) to identify the intraX instances of co-location (belonging to a clique) and the interX instances of co-location (belonging between two cliques) and then joins the intraX instances and the interX instances, respectively, to calculate the value of the participation index. This approach mines co-location patterns more efficiently than the join-based approach since it reduces a large number of join operations. The join-less approach uses the star neighborhood to identify spatial relationships. It is more efficient compared with the join-based approach and the partial-join approach since it generates co-location patterns without join operations.

In recent years, inspired by the transaction-based approach [6] and the transaction-free approach [7] in pattern mining, researchers have proposed many advanced/improved co-location mining approaches. The problem of mining co-location patterns with rare spatial events was first addressed by Huang et al. [8]. The concept of maximal clique has

been proposed and applied for co-location mining in the literature, for example, [9–11]. Yao et al. proposed an adaptive maximal co-location (AMCM) algorithm to address two limitations in the co-location mining method [12]. A multi-level method was developed to identify regional co-location patterns in two steps: first, global co-location patterns were detected and other non-prevalent co-location patterns were identified as candidates for regional co-location patterns; second, an adaptive spatial clustering method was applied to detect the sub-regions where regional co-location patterns are prevalent [13]. Some researchers have studied the relationship between the spatial co-location pattern and clustering [14–16], which provides a new idea for co-location mining algorithm. Ouyang et al. studied the co-location mining problem for fuzzy objects and 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), for mining co-location patterns at a membership threshold or within a membership range [17]. Zhou et al. have also made a lot of contributions to the co-location problem [18–20]. A novel clique-based approach for discovering complete and correct prevalent co-location patterns was proposed by Bao and Wang [21]. Celik et al. designed an indexing structure for co-location patterns and proposed algorithms (Zoloc-Miner) to discover zonal co-location patterns efficiently for dynamic parameters [22]. The concept of negative co-location patterns was defined by Jiang et al. [23]. Based on the analysis of the relationship between negative and positive participation indexes, they proposed methods for negative participation index calculation and negative pattern pruning strategies. Yu investigated a projection-based co-location pattern mining paradigm [24].

Although those approaches have solved the problem of co-location mining, they still have shortcomings—for example, the join-based approach requires a large number of join operations, building cliques is time consuming in the partial-join approach, and the computation time of the join-less approach increases with an increase in the co-location size. Most of these shortcomings are caused by instance join and excessive calculation of the participation index. For this reason, this paper presents a maximal instance algorithm for the fast mining of co-location patterns. The major contributions are the following:

- A maximal instance algorithm for mining co-location patterns is presented. This algorithm can generate row instances and co-locations without join operations, which can make co-location mining more efficient.
- The concept of maximal instance is introduced and used to generate all row instances of co-locations.
- Experiments show that the maximal instance algorithm is feasible and has better performance.

2. Maximal Instance Algorithm

In Figure 1a, each instance is uniquely identified by $T.i$, where T is the spatial feature type and i is the unique ID inside each spatial feature type, and the lines between instances represent neighbor relationships. Figure 1b shows the instances of co-location $\{A, B, C\}$ being generated by instance joins. A maximal instance is a row instance that does not appear as a subset of another row instance. In other words, maximal instances cannot combine with other instances to generate row instances of higher-size co-locations. For example, $\{A.1, B.2\}$ in Figure 1a is not a maximal instance since it is a subset of the row instance $\{A.1, B.2, C.1\}$, which in turn is a maximal instance. A maximal instance must be a maximal clique (a maximal clique is defined as a clique that does not appear as a subset of another clique in the same co-location pattern), but a maximal clique may not be a maximal instance since maximal cliques may not be row instances. Assuming $A.i$, $B.i$, and $B.j$ ($A.i$ is an instance of A , and $B.i$ and $B.j$ are instances of B) form a clique, i.e., they are neighbors to each other, $\{A.i, B.i, B.j\}$ is a maximal clique rather than a maximal instance as it is not a row instance. However, $\{A.1, B.2, C.1\}$ is a maximal instance and also a maximal clique.

Figure 1. (a) An example data set; (b) instance joins.

2.1. Generation of Row Instances

The generation of row instances is the basic part of co-location mining because only when all row instances are found can the participation index of co-locations be calculated. How to find row instances of co-locations is discussed here.

In the process of generating row instances of all co-locations, there are some differences between our algorithm and previous algorithms. The generation of row instances in the join-based algorithm, based on a join operation, can be represented by formula (1), and the detailed instance join process for the example data set (in Figure 1a) is shown in Figure 1b. The k -size row instances are generated from $k-1$ -size row instances, i.e., high-size row instances are generated by the join operation between low-size row instances. For example, the 2-row instance $\{A.1, B.2\}$ and $\{A.1, C.1\}$ can be joined to generate a 3-row instance $\{A.1, B.2, C.1\}$. This method for mining co-location is time consuming due to the large number of join operations required as the number of features and their instances increases.

$$E(RI_1) \xrightarrow{join} RI_2 \xrightarrow{join} \dots \xrightarrow{join} RI_k \quad (1)$$

Here, E is a set of spatial event types (i.e., RI_1) and RI_i is a set of row instances of i -size co-locations. In a join-based algorithm, the set of all row instances RI is a merge of $RI_1, RI_2, \dots$, and RI_k , i.e., $RI = \sum_{i=1}^k RI_i$. Every time the size of co-locations is increased, a join operation needs to be performed, resulting in $k-1$ being the number of times join operations are needed to generate a k -size row instance. The larger the k , the more the number of join operations needed and the more the execution time consumed. Therefore, the method (called the join-based method in this paper) of generating row instances by join operations is inefficient.

By reviewing the generation of row instances in the join-based algorithm, the relationship between a high-size row instance H generated from L and a low-size row instance L is obvious: L is a subset of H since H must contain the instances in L . In addition, if a high-size row instance exists, it must also exist for its corresponding low-size row instances. The corresponding low-size row instances are actually subsets of the high-size row instances. So, Lemma 1 can be obtained. According to Lemma 1, instead of using low-size row instances to generate high-size row instances, as in traditional algorithms, this paper thinks backward: using high-size row instances to generate low-size row instances.

Lemma 1. *An arbitrary non-empty subset of a k -size row instance must be a row instance of co-location.*

Proof of Lemma 1. This lemma is proved from the definition and generation process of row instances. (1) By definition, a row instance is essentially an R -proximity neighborhood, and the definition of R -proximity neighborhoods is given by Huang as follows: an R -proximity neighborhood is a set $I \subset S$ of instances that form a clique under the relation R . Accordingly, a set I is a row instance if any two instances in I satisfy the neighborhood relation. (2) Huang et al. gave a detailed introduction to the generation of row instances [1]. Similar to the generation of candidate co-locations, row instances are generated by a join strategy. An example in Figure 1 is used to illustrate the generation of row instances. Row instances $\{A.1, B.2\}$ and $\{A.1, C.1\}$ are joined to generate $\{A.1, B.2, C.1\}$. Then the neighbor relationship between $B.4$ and $C.1$. $\{A.1, B.2, C.1\}$ is a true row instance if $B.2$ has a neighbor relationship with $C.1$. On the contrary, $\{A.4, B.5, C.4\}$, generated by joining $\{A.4, B.5\}$ and $\{B.5, C.4\}$, fails to be a row instance since $A.4$ has no neighbor relationship with $C.4$. Thus, any two instances in a row instance satisfy the neighbor relationship. Let I_k be a set of instances in a k -size row instance. Because k -size row instances are generated from $k-1$ -size row instances, a conclusion is drawn: I_{k-1} is a proper subset of I_k (shown as the formula $I_{k-1} \subset I_k$). Therefore, it can be inferred that $I_1 \subset I_2 \subset \dots \subset I_k$. So, any set I_n is a proper subset of I_k ($1 \leq n < k$), i.e., $I_n \subset I_k$. If row instances are treated as sets, the following conclusion can be drawn: $RI_n \subset RI_k$. Here, RI_k is a set of k -size row instances and $1 \leq n < k$. In summary, an arbitrary non-empty subset of a k -size row instance must be a row instance of co-locations. $\square$

Accordingly, (1) a maximal instance is the maximal row instance for a co-location and it cannot join with another instance to generate a high-size row instance and (2) a further conclusion can be drawn from Lemma 1 that non-empty subsets of maximal instances are all row instances of co-locations. Once all maximal instances are found, all row instances can be generated by finding non-empty subsets of them. This method (shown in formula (2), $M = \{m_1, m_2, \dots, m_k\}$ is a set of maximal instances) for generating row instances does not require join operations, which can reduce a lot of computing time. The set of all row instances RI is a merge of non-empty subsets of maximal instances. Empty sets are not considered here since it is meaningless.

$$E \rightarrow M \xrightarrow{\text{gen_non-empty subset}} RI \quad (2)$$

A simple comparison is made to illustrate that this method is more efficient than the join-based method for generating row instances. The join-based method generates row instances by join operations. First, 2-size row instances are generated by joining spatial instances that have a neighbor relationship. Then high-size row instances are generated size by size until no higher-size row instances are generated. $k-1$ is the number of join operations required to generate all row instances, which is time consuming. Our method is to first generate the highest-size row instances (maximal instances) and then generate all row instances by finding non-empty subsets of the highest-size row instances, which requires no join operations and finding subsets only once. One obvious difference between the two methods is that the join-based method generates row instances from low sizes to high sizes and our method generates row instances from the highest size to low sizes. A large number of join operations are not needed in our method, which makes our method less time consuming.

Let k be the number of maximal instances, m_i represent a maximal instance ($1 \leq i \leq k$), $NS(m_i)$ represent the set of non-empty subsets of m_i , $\mathcal{B}(m_i)$ be the power set of m_i , $|m_i|$ be the number of instances in m_i , and $|RI|$ be the number of row instances. The two lemmas can be derived: $RI = \cup NS(m_i) = (\cup \mathcal{B}(m_i)) - \emptyset$; $|RI| = \sum_{i=1}^k 2^{|m_i|} - k$. The proofs are as follows. It should be noted that the universal set here is the set of spatial instances I .

Lemma 2. $RI = \cup NS(m_i) = (\cup \mathcal{B}(m_i)) - \emptyset$.

Proof of Lemma 2. First, according to the conclusion that non-empty subsets of maximal instances are all row instances of co-locations, there is $RI = \cup NS(m_i)$. Second, $NS(m_i) = \mathfrak{B}(m_i) - \emptyset$ since $\mathfrak{B}(m_i)$, the power set of m_i , is defined as a set of all subsets of m_i ; $NS(m_i)$ is the set of non-empty subsets of m_i . Therefore,

$$\begin{aligned} RI &= \cup NS(m_i) \\ &= \cup (\mathfrak{B}(m_i) - \emptyset) \\ &= \cup (\mathfrak{B}(m_i) - \emptyset) \\ &= \cup (\mathfrak{B}(m_i) \cap \sim \emptyset) \\ &= (\cup \mathfrak{B}(m_i)) \cap \sim \emptyset \\ &= (\cup \mathfrak{B}(m_i)) - \emptyset. \end{aligned}$$

□

Lemma 3. $|RI| = \sum_{i=1}^k 2^{|m_i|} - k$.

Proof of Lemma 3. For an n -set, the total number of subsets is $C_n^0 + C_n^1 + \dots + C_n^n = 2^n$, so $|\mathfrak{B}(m_i)| = 2^{|m_i|}$, and $|NS(m_i)| = |\mathfrak{B}(m_i)| - 1$ since $NS(m_i) = \mathfrak{B}(m_i) - \emptyset$. Therefore, $|RI| = \sum_{i=1}^k |NS(m_i)| = \sum_{i=1}^k (|\mathfrak{B}(m_i)| - 1) = \sum_{i=1}^k (2^{|m_i|} - 1) = \sum_{i=1}^k 2^{|m_i|} - k$. □

As shown in Table 1, $M = \{\{A.1, B.2, C.1\}, \{B.2, A.5\}, \{A.5, B.6\}, \{A.2, B.4\}, \{A.3, C.2\}, \{C.2, A.1\}, \{A.4, B.3, C.4\}, \{A.4, B.5\}, \{B.1, C.3\}\}$ is a set of maximal instances found from the example data set in Figure 1a. Row instances of all co-locations listed in the second column of Table 1 are non-empty subsets of maximal instances, and the number of row instances $|RI|$ can be calculated by Lemma 3.

$$\begin{aligned} |RI| &= (2^3 - 1) + (2^2 - 1) + (2^2 - 1) + (2^2 - 1) + (2^2 - 1) + (2^2 - 1) + (2^3 - 1) + (2^2 - 1) + (2^2 - 1) \\ &= (2^3 + 2^2 + 2^2 + 2^2 + 2^2 + 2^2 + 2^3 + 2^2 + 2^2) - 9 \\ &= 35. \end{aligned}$$

□

Table 1. Enumeration of the maximal instance and co-location instances.

M	Row Instances of All Co-Locations
$\{A.1, B.2, C.1\}$	$\{A.1\}, \{B.2\}, \{C.1\}, \{A.1, B.2\}, \{A.1, C.1\}, \{B.2, C.1\}, \{A.1, B.2, C.1\}$
$\{B.2, A.5\}$	$\{B.2\}, \{A.5\}, \{B.2, A.5\}$
$\{A.5, B.6\}$	$\{A.5\}, \{B.6\}, \{A.5, B.6\}$
$\{A.2, B.4\}$	$\{A.2\}, \{B.4\}, \{A.2, B.4\}$
$\{A.3, C.2\}$	$\{A.3\}, \{C.2\}, \{A.3, C.2\}$
$\{C.2, A.1\}$	$\{C.2\}, \{A.1\}, \{C.2, A.1\}$
$\{A.4, B.3, C.4\}$	$\{A.4\}, \{B.3\}, \{C.4\}, \{A.4, B.3\}, \{A.4, C.4\}, \{B.3, C.4\}, \{A.4, B.3, C.4\}$
$\{A.4, B.5\}$	$\{A.4\}, \{B.5\}, \{A.4, B.5\}$
$\{B.1, C.3\}$	$\{B.1\}, \{C.3\}, \{B.1, C.3\}$

2.2. RI-Tree Construction

As described in Section 2.1, maximal instances are used to generate the row instances of co-locations. The RI-tree for finding maximal instances is introduced in this section.

A maximal instance m should satisfy the following two conditions: (1) m is a row instance of a co-location and (2) m cannot join with other instances to generate a high-size row instance.

An RI-tree is a kind of rooted tree. The root of the RI-tree is labeled as F (F is a set of all spatial instances). A branch of the RI-tree is constructed as a corresponding connective sub-graph in the graph G . The node in the RI-tree represents the row instance of co-location patterns. The node u is the parent of the node v , and when v is a row instance of size k co-

locations, u is one of the row instances of size $k-1$ co-locations. An RI-tree is constructed to generate maximal instances, which is the key of co-location mining. As shown in Figure 2, a set of spatial instances with their spatial relationships (i.e., Euclidean distance) is input, and an algorithm implemented in Python outputs all maximal instances. In this process, neighbor relationships between instances of the same spatial feature type are not taken into consideration since our goal is to find the positive relationship between different spatial feature types.

```

Input:
  E: a set of spatial event types
  I: a set of instances  ⟨instance –
    id, spatial feature type, location⟩
  min_d: a distance threshold
  min_prev: a minimal prevalent threshold
Output:
  a set of maximal instances
Variables:
  M: a set of maximal instances
  R: a spatial neighbor relation
Method:
1.  R=gen_neighborhood;
2.  push all the instances in I into a stack  $RI_1$  in
    descending order;
3.  while (not empty  $RI_1$ ) do {
4.    i ← Pop an instance from  $RI_1$  and delete it in  $RI_1$ ;
5.    for (j in  $RI_1$ ) do {
6.      if ({i,j} in R) do {
7.        M.insert(0, T), T = {i,j}; }
8.      for (m in  $RI_1$ ) do {
9.        if ({T[1],m} in R or {m,T[1]} in R) do {
10.         if ({T[0],m} in R or {m,T[0]} in R) do {
11.           M.remove(T);
12.           T.insert(2, m);
13.           M.insert(0, T);
14.            $RI_1$ .remove(m); }
15.         }
16.       }
17.     }
18.   }

```

Figure 2. Algorithm for generating maximal instances.

Based on the definitions of RI-tree and maximal instance, there is the process of constructing an RI-tree.

Process of constructing an RI-tree:

- (1) Create the root of the RI-tree and label it as F .
- (2) Push all spatial instances into a set F in alphabetic and numerical descending order.
- (3) Pop an instance from RI_1 and delete it in RI_1 . Then create a child node of the root F for this instance.
- (4) Find out the instances that are neighbors of this instance, have different spatial features from this instance, and are bigger than this instance.
 - a. If not, return to (3).
 - b. If so, push them into a set T in alphabetic descending order and create a child node for them.
- (5) Find out the instances that are neighbors of the last instance in T and delete them in RI_1 .

- a. If not, push T in M ; return to (3).
 - b. If so and if they also have a neighborhood relationship with the rest of the instances in T , put them at the end of T and create a child node for them. Return to (5).
 - c. If so but if they are not all neighbors with other instances in T , push T in M , create another child node of the root for the last instance in T , and generate a new T by combining it with the last instance in T . Return to (5).
- (6) Repeat the operation above till RI_1 is empty.

Figure 3 shows the process of constructing an RI-tree for the example data set in Figure 1a. The leaf nodes of this RI-tree, such as $\{A.1, B.2, C.1\}$, $\{B.2, A.5\}$, $\{A.5, B.6\}$, $\{A.2, B.4\}$, $\{A.3, C.2\}$, $\{C.2, A.1\}$, $\{A.4, B.3, C.4\}$, $\{A.4, B.5\}$, and $\{B.1, C.3\}$, are maximal instances of the example data set. Let us take $\{A.1, B.2, C.1\}$ as an example to illustrate the gene ratio of a maximal instance. $A.1$ is a spatial instance popped from the example data set. Create a child node of $A.1$ for $A.1$ and $B.2$ because of the neighbor relationship between $A.1$ and $B.2$. Then find out the instance $C.1$ that has a neighbor relationship with $B.2$ and check whether there is a neighbor relationship between $A.1$ and $C.1$. A child node is created for $A.1, B.2$, and $C.1$ because of the neighbor relationship between $A.1$ and $C.1$. The spatial instance $D.3$ has a neighbor relationship with $C.1$ but not with $A.1$ and $B.2$, so $\{A.1, B.2, C.1\}$ is a maximal instance.

Figure 3. The process of constructing a row instance (RI)-tree.

2.2.1. Rules of an RI-Tree

1. Before building an RI-tree, all spatial instances are put into the set F . The instance will be deleted from F if a child node of the root is created for it or if it has a neighbor relationship with the child node of the root. For example, a child node of the root is created for $A.5$, so $A.5$ will be deleted from F ; $B.2$ and $C.1$ are deleted from F since they have neighbor relationships with $A.5$ ($A.5$ is a child node of the root).
2. Each node is a row instance of the co-location, and note that not all row instances are listed in the RI-tree but all neighbor relationships are contained in the RI-tree. The number of row instances is very large if there are a lot of instances in the spatial data. The aim of an RI-tree is to generate the maximal instance; there is no need to list all row instances in an RI-tree. A node in an RI-tree contains neighbor relationships between all instances in the node.
3. The nodes of the n -th layer of an RI-tree are $n-1$ -size row instances. The spatial instances in F are selected as the child node of the root, i.e., nodes in the second layer of the RI-tree. From layer 3 to layer n , there are size 2 row instances, size 3 row instances, $\dots$, and $n-1$ -size row instances. The highest size of the row instance of the example data in Figure 1 is 3 since the RI-tree constructed for this example data has only four layers.

4. After creating a child node of the root, scan its neighbor relationships. The neighbor relationships are identified by the Euclidean distance metric. Two instances have a neighbor relationship if their Euclidean distance is less than the user-defined minimum distance threshold [1]. If this node has a neighbor relationship with another instance, a child node of this node will be created under this branch for these two instances; if there is no common neighbor of instances in this node, another instance will be popped from F and a child node of the root will be created for it. For example, instance $B.2$ is a child node of the root and the node " $B.2, A.5$ " is created under this branch since $B.2$ has a neighbor relationship with $A.5$. There is no common neighbor of $B.2$ and $A.5$, but $A.5$ still has relationship with $B.6$, so a new child node of the root is created for $A.5$, and the node $A.5$ has " $A.5, B.6$ " as a child node.
5. The RI-tree is built from left to right. When a leaf node appears in one branch of the RI-tree, another branch can be established. A node is a leaf node if there is no common neighbor of instances in this node, which means the end of the branch where this node is located.
6. The leaf nodes of an RI-tree are maximal instances because a leaf node is a row instance in which all instances are neighbors and no other instance can join with it to generate a high-size row instance. It is obvious that the leaf node meets the conditions of maximal instance. The number of maximal instances equals the number of branches of the RI-tree.

2.2.2. Completeness of the RI-Tree

None of the neighbor relationships is omitted in the process of finding maximal instances. All the spatial instances are scanned, and all neighbor relationships between instances of different spatial types are considered in the process of constructing an RI-tree. Although not all neighbor relationships are shown in the RI-tree, they are considered when creating nodes. For example, the node " $A.1, B.2, C.1$ " contains the neighbor relationships between $A.1$ and $B.2$, $A.1$ and $C.1$, and $B.2$ and $C.1$, although the neighbor relationships between $A.1$ and $C.1$ on one hand and $B.2$ and $C.1$ on the other are not shown in the RI-tree. The neighbor relationships are also not duplicated. Once all neighbor relationships of an instance are identified, this instance will be deleted in F .

2.3. Generation of Co-Locations

Candidate co-location generation: The basic algorithms rely on a combinatorial approach and use `apriori_gen` [25] to generate size $k+1$ candidate co-locations from size- k prevalent co-locations. However, this paper generates candidate co-locations from maximal instances, which does not require any join operations. $C = \{e_1, e_2, \dots, e_k\}$ is a candidate co-location if $m = \{i_1, i_2, \dots, i_k\}$ is a maximal instance, i_n is an instance of spatial event types e_n , and $1 \leq n \leq k$. For example, $\{A.1, B.2, C.1\}$ is a maximal instance, so $T = \{A, B, C\}$ is a candidate co-location. Candidate co-locations are also generated from candidate co-locations that are not prevalent. Candidate co-locations generated from maximal instances are not all prevalent. Candidate co-locations that fail to be a prevalent pattern are selected, and their subsets are corresponding low-size candidate co-locations. That is because of the anti-monotonicity of the participation index. If $pi(C) < min_prev$ (C is a candidate co-location, and $pi(C)$ is the participation index of C), there must exist $pi(c) \geq pi(C)$ (c is a subset of C) and c may be a prevalent co-location, so c is a candidate co-location. Taking Figure 1 as an example, if min_prev is set to 0.4, $T = \{A, B, C\}$ is not a prevalent co-location since $pi(T) = \min\{pr(T, A), pr(T, B), pr(T, C)\} = \min(2/5, 2/6, 2/4) = 1/3 < 0.4$, where pr is the participation ratio. Therefore, $\{A, B\}$, $\{A, C\}$, and $\{B, C\}$ (subsets of $\{A, B, C\}$) are candidate co-locations.

Pruning: To make our algorithm more efficient, this step introduces the pruning strategies that can greatly reduce the unnecessary calculation time. Prevalence-based pruning [1] is also used in our algorithm. The refinement filtering of co-locations is done by the participation index values calculated from their co-location instances. Prevalent

co-locations satisfying a given threshold are selected. Actually, there is no need to calculate the participation index of each candidate co-location. Because the participation ratio and the participation index are antimonotone (monotonically nonincreasing) as the size of the co-location increases [1]), if a high-size candidate co-location is prevalent, subsets of this candidate co-location are all prevalent. Therefore, participation indexes of high-size candidate co-locations (generated by maximal instances) are prioritized. If the participation index values are above a given threshold, there is no need to calculate the participation indexes of subsets of a high-size candidate co-location; if the participation index values are below a given threshold, the participation indexes of the subsets of high-size candidate co-locations need to be calculated.

The second pruning strategy can be explained by Lemma 4.

Lemma 4. *Subsets of a candidate co-location are prevalent if it is a prevalent co-location.*

Proof of Lemma 4. This lemma is proved by the antimonotone property of the participation index. C is a candidate co-location; C' is a subset of C ; and k and n ($k > n$) are the number of instances in C and the number of instances in C' , respectively. From the above formula, it can be concluded that the participation index of a candidate co-location is less than or equal to that of its subset. If $pi(C) < min_prev$, then $pi(C') < min_prev$ must be established, so the following conclusion can be drawn: subsets of a candidate co-location are prevalent if it is a prevalent co-location.

$$\begin{aligned}
 pi(C) &= \min \left\{ \frac{|\pi_{e_1}(table_instance(C))|}{|e_1|}, \dots, \right. \\
 &\quad \left. \frac{|\pi_{e_k}(table_instance(C))|}{|e_k|} \right\} \\
 &= \min_{i=1}^k \{pr(C, e_i)\} \\
 &\leq \min_{i=1}^n \{pr(C, e_i)\} \\
 &\leq \min_{i=1}^n \{pr(C', e_i)\} = pi(C').
 \end{aligned}$$

□

Quite a lot of maximal instances are generated in the process of mining co-locations from the real data set introduced in Section 4. $\{A, I, T, P, R\}$ is a candidate co-location since there are row instances of $\{A, I, T, P, R\}$ in maximal instances, and $\{A, I, T, P, R\}$ is the highest-size candidate co-location since there are only five facility types in the real data set. The participation index of $\{A, I, T, P, R\}$ is calculated first. When min_d is set to 1, $pi(\{A, I, T, P, R\}) = 0.1$. $\{A, I, T, P, R\}$ fails to be a prevalent co-location if min_prev is set to 0.4. Therefore, $\{A, I, T, P\}$, $\{A, I, T, R\}$, $\{A, I, P, R\}$, $\{A, T, P, R\}$, and $\{I, T, P, R\}$, which are the subsets of $\{A, I, T, P, R\}$, become candidate co-locations. In the subsets of $\{A, I, T, P, R\}$, those row instances that contain four instances are row instances of size 4 co-locations. The size 4 maximal instances are also row instances of size 4 co-locations. By calculating their participation indexes, it is clear that $\{A, T, P, R\}$ is a prevalent co-location with $pi(\{A, T, P, R\}) = 0.48$. According to the anti-monotonicity of the participation index, the participation indexes of $\{A, T, P\}$, $\{A, T, R\}$, $\{A, P, R\}$, and $\{T, P, R\}$ must be greater than 0.48. $\{A, T, P\}$, $\{A, T, R\}$, $\{A, P, R\}$, and $\{T, P, R\}$ are prevalent co-locations, so there is no need to calculate their participation indexes, which saves a lot of time. This is the pruning strategy mentioned above.

2.4. Discussions for Maximal Instance Algorithm

The idea of the presented algorithm is different from the existing algorithms. The differences are as follows:

- The existing methods identify high-size row instances from low-size row instances, but the presented algorithm first identifies the highest-size row instances (maximal instances) based on the relationship of spatial instances and then generates all low-size row instances from highest-size row instances.
- The existing methods generate candidate patterns by joining row instances, but the presented algorithm generates candidate patterns from maximal instances without join operations.

2.4.1. Comparison Analysis of Row Instance Generation

The reason why our method for generating all row instances is more efficient than the join-based algorithm is demonstrated in detail here. Let $T_{gen_row\ instance}$ and $T'_{gen_row\ instance}$ be the computation cost of generating all row instances by the join-based algorithm and our method. The following equations show the computation cost functions:

$$T_{gen_row\ instance} = T_{scan(neighobr\ relationship)} + (k-1) \times T_{join} \quad (3)$$

$$T'_{gen_row\ instance} = T_{gen_m} + T_{gen_instance} \quad (4)$$

T_{join} represents the computation cost of a join operation, T_{gen_m} represents the computation cost of generating all maximal instances, and $T_{gen_instance}$ represents the computation cost of finding the subsets of maximal instances. $T_{gen_m} \approx T_{scan(neighobr\ relationship)}$ because generating maximal instances is also a scanning process of neighbor relationship and $T_{gen_instance} \ll (k-1) \times T_{join}$.

Therefore,

$$\begin{aligned} \frac{T_{gen_row\ instance}}{T'_{gen_row\ instance}} &= \frac{T_{scan(neighobr\ relationship)} + (k-1) \times T_{join}}{T_{gen_m} + T_{gen_instance}} \\ &\approx \frac{T_{scan(neighobr\ relationship)} + (k-1) \times T_{join}}{T_{scan(neighobr\ relationship)} + T_{gen_instance}} \\ &> 1. \end{aligned}$$

Apparently, $T_{gen_row\ instance} > T'_{gen_row\ instance}$. So the conclusion can be drawn that our method of generating all row instances is more efficient than the join-based algorithm.

2.4.2. Comparison Analysis of Maximal Instance Algorithm

The computational cost of our algorithm with three basic algorithms is compared here. Each algorithm has an initial step of materializing neighborhoods. The join-based algorithm first gathers the neighbor pairs per candidate co-location, the partial-join algorithm first generates the disjoint clique neighborhoods using a simple grid partitioning, the join-less algorithm first generates the star neighborhoods, and our algorithm first generates maximal instances. S represents a spatial data set; $T_{neighbor_pairs}(S)$, $T_{clique_neighborhoods\&cut_relations}(S)$, $T_{star_neighborhoods}(S)$, and $T_{gen_m}(S)$ represent the computational costs of the initial steps in each algorithm; and $T_{jb}(2)$, $T_{pj}(2)$, and $T_{jl}(2)$ represent the cost of generating size 2 co-locations in the join-less algorithm, the partial-join algorithm, and the join-based algorithm, respectively. Let T , T_{jl} , T_{pj} , and T_{jb} be the total computation cost of our algorithm, the join-less algorithm, the partial-join algorithm, and the join-based algorithm, respectively. The following equations show the total cost functions:

$$T = T_{gen_m}(S) + T_{gen_instance},$$

$$T_{jl} = T_{star_neighborhoods}(S) + T_{jl}(2) + \sum_{k>2} T_{jl}(k),$$

$$T_{pj} = T_{clique_neighborhoods\&cut_relations}(S) + T_{pj}(2) + \sum_{k>2} T_{pj}(k),$$

$$T_{jb} = T_{neighbor_pairs}(S) + T_{jb}(2) + \sum_{k>2} T_{jb}(k),$$

Here, k is greater than 2.

Yoo and Shekhar [5] proposed the following comparative relationships:

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

because (1) the join-less algorithm involves an additional cost to generate the star neighborhoods from the neighbor pairs compared with join-based algorithm and (2) an additional cost is incurred when finding all size 2 inter-instances with cut relationships in the partial-join algorithm, the overall cost is expected to be a little greater than the cost to generate the star neighborhood set.

Lemma 5. $T_{star_neighborhoods}(S) > T_{gen_m}(S)$.

Proof of Lemma 5. The star neighborhood partition model in the join-less algorithm takes each spatial object as a central object and finds out the remaining objects in its neighborhood. Thus, this process is performed n times, where n is the number of spatial objects. The process of generating all maximal instances needs to be executed $|C_m|$ times, where $|C_m|$ is the number of maximal instances. Apparently, $n > |C_m|$, so the star neighborhood partition model in the join-less method is more time consuming than the process of generating the maximal instance in our method.

$$\begin{aligned} \frac{T_{star_neighborhoods}(S)}{T_{gen_m}(S)} &\approx \frac{n \times t_{scan(neighbor\ relationship)}}{|C_m| \times t_{scan(neighbor\ relationship)}} \\ &= \frac{n}{|C_m|} \\ &> 1. \end{aligned}$$

□

Lemma 6. $T_{gen_m}(S) > T_{neighbor_pairs}(S) + T_{jb}(2)$.

Proof of Lemma 6. Gathering the neighbor pairs per candidate co-location is the most basic operation in co-location mining. The neighbor pairs are used to generate maximal instances, and it involves additional cost compared with gathering the neighbor pairs per candidate co-location and calculating their prevalence measures. □

According to Lemma 5 and Lemma 6, the following equation can be reached:

$$\begin{aligned} T_{clique_neighborhoods\&cut_relations}(S) + T_{pj}(2) \\ > T_{star_neighborhoods}(S) + T_{jl}(2) \\ &> T_{gen_m}(S) \\ &> T_{neighbor_pairs}(S) + T_{jb}(2). \end{aligned} \tag{5}$$

The advantage of the maximal instance algorithm lies in the latter part. Three basic algorithms generate bigger-size co-locations size by size after generating size 2 co-locations. This operation needs to be repeated $k-2$ times to generate all co-locations. No operation can be omitted, and a cross-size operation is not allowed (size k co-locations cannot be directly generated from size $k-2$ co-locations; size k co-locations are generated from size $k-1$ co-locations after generating size $k-1$ co-locations from size $k-2$ co-locations). Therefore, the execution time of each join operation should be added to the total computational costs of the three basic algorithms.

However, once the maximal instance algorithm generates maximal instances, row instances of all size co-locations can be generated from the subsets of maximal instances. There is no need for a join operation. A large amount of time consumption is avoided in this part; the bigger the size of the co-locations, the more obvious the effect, and thus the following equations:

$$T_{gen_instance} \ll \sum_{k>2} T_{jl}(k), \quad (6)$$

$$T_{gen_instance} \ll \sum_{k>2} T_{pj}(k), \quad (7)$$

$$T_{gen_instance} \ll \sum_{k>2} T_{jb}(k). \quad (8)$$

On the basis of Equations (5)–(8), we can say that our algorithm has a lower computational cost than the three basic algorithms.

$$T < T_{jl},$$

$$T < T_{pj},$$

$$T < T_{jb}.$$

2.5. Overview of the Maximal Instance Algorithm

This section introduces the steps in the maximal instance algorithm. As shown in Figure 4, this algorithm has four steps in the flow chart. The participation index is still used as the prevalence measure to determine whether a co-location pattern is prevalent. As shown in Figure 5, the maximal instance algorithm takes a set E of spatial event types, a set I of instances, a distance threshold min_d , and a minimal prevalent threshold min_prev . The algorithm outputs a correct set of co-location patterns with participation index $> min_prev$. The set C_1 of candidate size 1 co-locations and the set P_1 of prevalent size 1 co-locations are initialized to E , since the value of the participate index is 1 for all size 1 co-locations. The set RI_1 of row instances of size 1 co-locations is initialized to I .

Figure 4. The flow of the maximal instance algorithm.

```

Input:
  E: a set of spatial event types
  I: a set of instances (instance –
    id, spatial feature type, location)
  min_d: a distance threshold
  min_prev: a minimal prevalent threshold
Output:
  M: a set of maximal instances
    a correct set of co-location patterns with
    participation index > min_prev.
Variables:
  k: co-location size
  RI: a set of row instances of all co-locations
  RIk: a set of row instances of size k co-locations
  Ck: a set of candidate size k co-locations
  Pk: a set of prevalent size k co-location
  NPk: a set of not prevalent size k co-location
Method:
1.  k=1, C1 = E, P1 = E, RI1 = I;
2.  M = gen_maximal_instance(R, E)
3.  RI = gen_subset(M);
4.  while (not empty Pk and k > 2) do {
5.    k = len(mi);
6.    Ck = gen_candidate_co – location(M);
7.    Pk = select_prevalent_co –
    location(Ck, RIk, min_prev);
8.    NPk = select_not_prevalent_co –
    location(Ck, RIk, min_prev);
9.    Ck-1 = gen_subset(NPk);
10. }
11. return U(P1, ..., Pk).

```

Figure 5. Algorithm for generating co-location patterns.

Step 1 (maximal instance generation): Our algorithm aims to find the maximal instances first, which is the key to co-location mining. A set of spatial instances with their spatial relationships (i.e., Euclidean distance) is input, each instance is a vector instance – id, location, spatial feature type, and an algorithm implemented in Python outputs all maximal instances. In this process, the neighbor relationships between instances of the same spatial feature type are not taken into consideration since our goal is to find the positive relationship between different spatial feature types.

Step 2 (row instance generation): Row instances of co-locations are generated by finding subsets of maximal instances instead of join operations between row instances.

Step 3 (candidate co-location generation): $C = \{e_1, e_2, \dots, e_k\}$ is a candidate co-location if $m = \{i_1, i_2, \dots, i_k\}$ is a maximal instance, i_n is an instance of spatial event types e_n , and $1 \leq n \leq k$. For example, $\{A.1, B.2, C.1\}$ is a maximal instance, so $T = \{A, B, C\}$ is a candidate co-location, and a candidate co-location is also generated from candidate co-locations that are not prevalent.

Step 4 (pruning): There are two pruning strategies in the maximal instance algorithm: (1) prevalence-based pruning is also used in our algorithm, and (2) participation indexes of high-size candidate co-locations (generated by maximal instances) are prioritized. If the participation index values are above a given threshold, there is no need to calculate the participation indexes of subsets of the high-size candidate co-location; if the participation index values are below a given threshold, the participation indexes of subsets of high-size candidate co-locations need to be calculated.

3. Experiments

In this section, experiments are carried with synthetic data sets and a real data set to evaluate our algorithm.

3.1. Synthetic Data Set

The synthetic data sets are generated in Python, as shown in Figure 6. There are 60 different spatial event types in a synthetic data set. The number of instances of each spatial feature stored in the set Value is a random number between 5 and 10. E is a set of all spatial event types, and I is a set of instances. Each instance is a point distributed in a spatial framework $D_1 \times D_2$ (150×100), and these point data have their own unique coordinates. The neighbor relationships of the synthetic data sets are generated by calculating the Euclidean distance of those points and stored in the set R .

Figure 6. Generation process of a synthetic data set.

3.2. Real Data Set

The real data set is about college and university locations, hospital locations, library locations, nursing home locations from Maine E911 address data from MainePUC, and point locations of the mineral collecting sites in Maine [26]. Each event in this data set with a different spatial feature types has a unique event ID. Contained in each event is county, town, latitude, longitude, and other information. Location information with only latitude and longitude is considered in this experiment. The event numbers of spatial feature types are shown in Table 2.

Table 2. Description of event numbers of spatial feature types.

Spatial Feature Type	Abbreviation	Number of Events
College and University	U	286
Hospital	H	74
Library	L	83
Nursing Home	N	45
Mineral Collecting Site	M	32

3.3. Experiment Results

A summary of our experiment result is presented here. A real data set of college and university locations, hospital locations, library locations, nursing home locations from Maine E911 address data from MainePUC, and point locations of the mineral collecting sites in Maine is studied in this paper. The longitude and latitude of each facility are used as coordinates to calculate the Euclidean distance. The effects of min_d on the number of neighbor relationships, participation ratio, and participation index are evaluated. Table 2 shows the number of events for each spatial feature type in the real data set.

Table 3 shows the participation indexes of co-location patterns. When min_d is set to 0.4 and min_prev is set to 0.5, co-location patterns $\{U, H\}$, $\{U, L\}$, $\{H, L\}$, $\{U, H, L\}$, $\{U, H, N\}$, and $\{U, L, N\}$ are prevalent and their participation indexes are 0.61, 0.63, 0.81, 0.59, 0.53, and 0.73, respectively. The co-location $\{U, H\}$ means hospitals are frequently located together in a geographic space with colleges and universities. For example, the University of Southern Maine is adjacent to two hospitals in the south, namely Maine

Medical Center and Northern Light Mercy Hospital. The co-location $\{U, L\}$ means libraries are frequently located together in a geographic space with colleges and universities. This is mainly because some libraries are located inside universities and colleges. For example, the largest library in Maine, the Fogler Library, is located in the University of Maine.

Table 3. The participation indexes of co-location patterns.

Co-Location Patterns	Participation Indexes (<i>min_d</i> Is Set to 0.4)
$\{U, H\}$	0.61
$\{U, L\}$	0.63
$\{H, L\}$	0.81
$\{U, H, L\}$	0.59
$\{U, H, N\}$	0.53
$\{U, H, M\}$	0.44
$\{U, L, N\}$	0.73
$\{U, L, M\}$	0.47
$\{U, N, M\}$	0.41

The result quality comparison

Firstly, the quality of the generated maximal instance is tested in terms of the total number of maximal instances, the number of maximal instances per size, and the number of maximal instances with errors and repetitions. The quality result of the maximal instances is shown in the Table 4. There are 1342 maximal instances in total. After removing the repetition, the number of size 2, size 3, size 4, and size 5 maximal instances is 1268, 17, 5, and 3, respectively. These 1342 maximal instances are not all correct. Among them, seven maximal instances with a size greater than 5 are errors. The maximum size of maximal instances is 5 since there are only five feature types in the real data set. Moreover, there are 42 repeated maximal instances in the size 2 maximal instances, but this phenomenon does not appear in the other higher-size maximal instances. It can be seen that the quality of the generated maximal instances is relatively good. These maximal instances of errors and repetitions have little effect on the experimental results.

Table 4. The quality of maximum instances.

	Number
Maximal Instances	1342
Size 2 Maximal Instances	1268
Size 3 Maximal Instances	17
Size 4 Maximal Instances	5
Size 5 Maximal Instances	3
False	7
Repetitive	42

Figure 7 shows a comparison of row instances generated by the join-based algorithm and the maximal instance algorithm. Row instance is the basis of co-location participation index. Only when the number of row instances is clear can the participation index of co-locations be calculated and whether co-locations are frequent be judged. Therefore, this paper used the join-based algorithm and the maximal instance algorithm to generate all the row instances, recorded the number of row instances in each size, and made a comparison. The results show that the number of size 3, size 4, and size 5 row instances generated by the two algorithms is the same and only the number of size 2 row instances is slightly different. The maximal instance algorithm generates 42 more size 2 row instances than the join-based algorithm, which is because the maximal instance algorithm generates 42 repeated maximal instances in the size 2 maximal instances. The impact of these 42 row instances on participation is very small.

Figure 7. Comparison of row instances generated by the join-based algorithm a and the maximal instance algorithm.

On comparing the results, it is clear that the maximal instance algorithm only has a slight difference in the size 2 instance, which will not have a great impact on the quality of the generated co-locations.

Effect of min_d on the number of neighbor relationships

The effects of min_d on the number of neighbor relationships are evaluated. The value of min_d is changed while the other experimental parameters remain unchanged. The number of neighbor relationships between any two spatial feature types is checked successively. The result is shown in Figure 8. In general, the number of neighbor relationships between any two types increases with an increase in the value of min_d . With the increase in min_d , the numbers of neighbor relationships of $\{U, H\}$ and $\{U, L\}$ are far more than that of other patterns. Because of the large number of events of type U, H , and L , the numbers of neighbor relationships of $\{U, H\}$ and $\{U, L\}$ increase sharply to more than 7000 when min_d increases to 0.9.

(a)

(b)

Figure 8. (a) The number of neighbor relationships of $\{U, L\}$, $\{U, M\}$, $\{U, N\}$, $\{U, H\}$ and $\{H, L\}$; (b) the number of neighbor relationships of $\{H, N\}$, $\{H, M\}$, $\{L, M\}$, $\{L, N\}$ and $\{M, N\}$.

Effect of min_d on participation ratio and participation index

The effects of min_d on participation ratio and participation index are evaluated. The value of the parameter min_d is changed to observe the values of the participation ratio and the participation index of co-locations. Figure 9 shows the change in the participa-

tion ratio of co-locations $\{U, H, L\}$, $\{U, H, N\}$, $\{U, H, M\}$, $\{U, L, M\}$, $\{U, L, N\}$, and $\{U, N, M\}$. It is obvious that the values of the participation ratio increase with an increase in min_d . The changes in the participation indexes of different co-locations are shown in Figure 10a,b. The following observations can be made:

Figure 9. (a) Pr of co-location $\{U, H, L\}$; (b) Pr of co-location $\{U, H, N\}$; (c) Pr of co-location $\{U, H, M\}$; (d) Pr of co-location $\{U, L, M\}$; (e) Pr of co-location $\{U, L, N\}$; (f) Pr of co-location $\{U, N, M\}$.

Figure 10. (a) Pi of $\{U, H, L\}$, $\{U, H\}$, $\{U, L\}$ and $\{H, L\}$; (b) Pi of $\{U, H, N\}$, $\{U, H\}$, $\{U, N\}$ and $\{H, N\}$.

- (1) $pi(\{U, H, L\})$ (the value of the participation index of $\{U, H, L\}$) is always smaller than $pi(\{U, H\})$, $pi(\{U, L\})$, and $pi(\{H, L\})$, and the same is the case in Figure 10b, where $pi(\{U, H, N\})$ is always smaller than $pi(\{U, H\})$, $pi(\{U, N\})$, and $pi(\{H, N\})$. This is because the participation ratio and the participation index are antimonotone (monotonically nonincreasing) as the size of the co-location increases [1] and it shows that our results are correct.
- (2) The combination of the minimum values of the three curves in Figure 9a is the curve of $pi(\{U, H, L\})$ in Figure 10a, since the $pi(C)$ of a co-location $C = \{e_1, \dots, e_k\}$ is defined as $\min_{i=1}^k \{pr(C, e_i)\}$.

3.4. Experimental Evaluation

The synthetic data sets are used to check whether our algorithm has better performance. Two comparative experiments on running time were done to evaluate the performance of the maximal instance algorithm. As shown in Figure 11, the running time of generating row instances is compared when min_d changes. The time to generate maximal instances is used to indicate the time to generate row instances of the maximal instance algorithm, because once maximal instances are found, row instances are found. The result shows that the running time increases with an increase in min_d and the maximal instance algorithm consumes less time. Figure 12 shows the effect of the number of spatial instances on the running time, and no matter how the number of spatial instances changes, the maximal instance algorithm still consumes less time. Obviously, the maximal instance algorithm has better performance in the process of generating prevalent co-locations since there is no need for join operations in the maximal instance algorithm. Therefore, it can be concluded that the maximal instance algorithm is more efficient than the three basic algorithms.

Figure 11. The effect of min_d on the running time.

Figure 12. The effect of the number of spatial instances on the running time.

4. Conclusions and Future Work

This paper proposes a new concept of maximal instances that is fundamental to our work and proves that generating row instances by maximal instances can make sense. A comparison with generating row instances by join operations proves that generating row instances by maximal instances is more efficient. An RI-tree is constructed to find maximal instances from a spatial data set. The proposed algorithm, the maximal instance algorithm, is based on maximal instances to mine co-location patterns. Since join operations are needed in this algorithm, it can reduce a lot of computing time. A detailed comparison with the three basic algorithms (join-based algorithm, partial-join algorithm, join-less algorithm) shows the advantage of our algorithm in terms of time consumption.

Synthetic data sets and a real data set are used to examine the effect of our algorithm. The real data set is about college and university locations, hospital locations, library locations, nursing home locations from Maine E911 address data from MainePUC, and point locations of the mineral collecting sites in Maine. This data set has five spatial feature types: college and university, hospital, library, nursing home, and mineral collecting site. The prevalent co-locations $\{U, H\}$, $\{U, L\}$, $\{H, L\}$, $\{U, H, L\}$, $\{U, H, N\}$, and $\{U, L, N\}$ are generated when min_d is set to 0.4 and min_prev is set to 0.5 ($pi(\{U, H\}) = 0.61$, $pi(\{U, L\}) = 0.63$, $pi(\{H, L\}) = 0.81$, $pi(\{U, H, L\}) = 0.59$, $pi(\{U, H, N\}) = 0.53$, $pi(\{U, L, N\}) = 0.73$). The participation indexes of the co-locations vary with a change in the experimental parameter min_d and satisfy anti-monotonicity, and comparative experiments on running time show that the maximal instance algorithm has better performance.

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

Author Contributions: Conceptualization, G.Z.; methodology, Q.L.; software, Q.L.; validation, Q.L.; formal analysis, Q.L.; investigation, G.Z., Q.L. and G.D.; resources, G.Z.; data curation, Q.L.; writing—original draft preparation, Q.L.; writing—review and editing, G.Z. and G.D.; visualization, Q.L.; supervision, G.Z. and G.D.; project administration, G.Z.; funding acquisition, G.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science of China under grant numbers 41431179 and 41961065; the Guangxi Innovative Development Grand Grant under grant numbers GuikeAA18118038 and GuikeAA18242048; and the National Key Research and Development Program of China under grant number 2016YFB0502501 and the BaGuiScholars Program of Guangxi.

Data Availability Statement: Data available in a publicly accessible repository.

Conflicts of Interest: The authors declare no conflict of interest.

References

- Huang, Y.; Shekhar, S.; Xiong, H. Discovering colocation patterns from spatial data sets: A general approach. *IEEE Trans. Knowl. Data Eng.* **2004**, *16*, 1472–1485. [\[CrossRef\]](#)
- Shekhar, S.; Huang, Y. Discovering Spatial co-location patterns: A summary of results. In *Advances in Spatial and Temporal Databases*; Jensen, C.S., Schneider, M., Seeger, B., Tsotras, V.J., Eds.; Springer: Berlin/Heidelberg, Germany, 2001; p. 2121.
- Yoo, J.S.; Shekhar, S.; Smith, J.; Kumquat, J.P. A partial join approach for mining co-location patterns. In Proceedings of the 12th annual ACM international workshop on Geographic information systems—GIS '04, Washington, DC, USA, 12–13 November 2004; pp. 241–249. [\[CrossRef\]](#)
- Yoo, J.S.; Shekhar, S.; Celik, M. A Join-Less Approach for Co-Location Pattern Mining: A Summary of Results. In Proceedings of the Fifth IEEE International Conference on Data Mining (ICDM'05), Houston, TX, USA, 27–30 November 2005; p. 4.
- Yoo, J.S.; Shekhar, S. A join-less approach for mining spatial co-location pattern. *IEEE Trans. Knowl. Data Eng.* **2006**, *18*, 1323–1337.
- Djenouri, Y.; Lin, J.C.-W.; Nørnvåg, K.; Ramampiaro, H. Highly Efficient Pattern Mining Based on Transaction Decomposition. In Proceedings of the 2019 IEEE 35th International Conference on Data Engineering (ICDE), Macao, China, 8–11 April 2019; pp. 1646–1649.
- Zhang, B.; Lin, J.C.-W.; Shao, Y.; Fournier-Viger, P.; Djenouri, Y. Maintenance of Discovered High Average-Utility Itemsets in Dynamic Databases. *Appl. Sci.* **2018**, *8*, 769. [\[CrossRef\]](#)
- Huang, Y.; Pei, J.; Xiong, H. Mining Co-Location Patterns with Rare Events from Spatial Data Sets. *GeoInformatica* **2006**, *10*, 239–260. [\[CrossRef\]](#)
- Verhein, F.; Al-Naymat, G. Fast Mining of Complex Spatial Co-location Patterns Using GLIMIT. In Proceedings of the Seventh IEEE International Conference on Data Mining Workshops (ICDMW 2007), Omaha, NE, USA, 28–31 October 2007; pp. 679–684.
- Al-Naymat, G. Enumeration of maximal clique for mining spatial co-location patterns. In Proceedings of the 2008 IEEE/ACS International Conference on Computer Systems and Applications, Doha, Qatar, 31 March–4 April 2008; pp. 126–133.
- Kim, S.K.; Kim, Y.; Kim, U. Maximal cliques generating algorithm for spatial colocation pattern mining. In *Secure and Trust Computing, Data Management and Applications*; Park, J.J., Lopez, J., Yeo, S.S., Shon, T., Taniar, D., Eds.; Springer: Berlin/Heidelberg, Germany, 2011; Volume 186, pp. 241–250.
- Yao, X.; Wang, D.; Peng, L.; Chi, T. An adaptive maximal co-location mining algorithm. In Proceedings of the 2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), Fort Worth, TX, USA, 23–28 July 2017; pp. 5551–5554. [\[CrossRef\]](#)
- Deng, M.; Cai, J.; Liu, Q.; He, Z.; Tang, J. Multi-level method for discovery of regional co-location patterns. *Int. J. Geogr. Inf. Sci.* **2017**, *31*, 1–25. [\[CrossRef\]](#)
- Huang, Y.; Zhang, P.; Zhang, C. On the Relationships between Clustering and Spatial Co-Location Pattern Mining. *Int. J. Artif. Intell. Tools* **2008**, *17*, 55–70. [\[CrossRef\]](#)
- Jiamthapthaksin, R.; Eick, C.F.; Vilalta, R. A framework for multi-objective clustering and its application to co-location mining. In *Advanced Data Mining and Applications*; Huang, R., Yang, Q., Pei, J., Gama, J., Meng, X., Li, X., Eds.; Springer: Berlin/Heidelberg, Germany, 2009; Volume 5678, pp. 188–199.
- Yu, W. Identifying and Analyzing the Prevalent Regions of a Co-Location Pattern Using Polygons Clustering Approach. *ISPRS Int. J. Geo-Inf.* **2017**, *6*, 259. [\[CrossRef\]](#)
- Ouyang, Z.; Wang, L.; Wu, P. Spatial Co-Location Pattern Discovery from Fuzzy Objects. *Int. J. Artif. Intell. Tools* **2017**, *26*. [\[CrossRef\]](#)
- Zhou, G.; Wang, L. Co-location decision tree for enhancing decision-making of pavement maintenance and rehabilitation. *Transp. Res. Part C: Emerg. Technol.* **2012**, *21*, 287–305. [\[CrossRef\]](#)
- Zhou, G.; Zhang, R.; Zhang, D. Manifold Learning Co-Location Decision Tree for Remotely Sensed Imagery Classification. *Remote Sens.* **2016**, *8*, 855. [\[CrossRef\]](#)
- Zhou, G.; Li, Q.; Deng, G.; Yue, T.; Zhou, X. Mining Co-Location Patterns with Clustering Items from Spatial Data Sets. *ISPRS-Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2018**, *XLIII-3*, 2505–2509. [\[CrossRef\]](#)
- Bao, X.; Wang, L. A clique-based approach for co-location pattern mining. *Inf. Sci.* **2019**, *490*, 244–264. [\[CrossRef\]](#)
- Celik, M.; Kang, J.M.; Shekhar, S. Zonal co-location pattern discovery with dynamic parameters. In Proceedings of the Seventh IEEE International Conference on Data Mining, Omaha, NE, USA, 28–31 October 2007.
- Jiang, Y.; Wang, L.; Chen, H. Discovering both positive and negative co-location rules from spatial data sets. Software Engineering and Data Mining (SEDM). In Proceedings of the 2010 2nd International Conference on IEEE, Chengdu, China, 23–25 June 2010.
- Yu, P. *FP-Tree based Spatial Co-Location Pattern Mining*; University of North Texas: Denton, TX, USA, 2005.
- Agrawal, R.; Srikant, R. Fast algorithms for mining association rules. In *Readings in Database Systems*, 3rd ed.; Stonebraker, M., Hellerstein, J.M., Eds.; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 1998; pp. 580–592.
- ArcGIS Hub. Available online: <https://hub.arcgis.com/> (accessed on 20 February 2021).