Efficient Discovery of Spatial Co-Location Patterns Using the iCPI-tree

Lizhen Wang*,[a], Yuzhen Bao[a] and Zhongyu Lu[b]

[a]*Department of Computer Science and Engineering, School of Information, Yunnan University, Kunming, 650091, P. R. China*

[b]*Department of Informatics, School of Computing and Engineering, University of Huddersfield, Huddersfield, UK, HD1 3DH*

Abstract: With the rapid growth and extensive applications of the spatial dataset, it's getting more important to solve how to find spatial knowledge automatically from spatial datasets. Spatial co-location patterns represent the subsets of features whose instances are frequently located together in geographic space. It's difficult to discovery 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 identifying the table instances of co-location patterns. The essence of co-location patterns discovery and four co-location patterns mining algorithms proposed in recent years are analyzed, and a new join-less approach for co-location patterns mining, which based on a data structure----iCPI-tree (Improved Co-location Pattern Instance Tree), is proposed. The iCPI-tree is an improved version of the CPI-tree which materializes spatial neighbor relationships in order to accelerate the process of identifying co-location instances. This paper proves the correctness and completeness of the new approach. Finally, an experimental evaluations using synthetic and real world datasets show that the algorithm is computationally more efficient.

Keywords: Spatial data mining, co-location rules, table instances (or co-location instances), the iCPI-tree.

1. INTRODUCTION

Spatial data mining is the process to discover interesting and previous unknown, but potential useful patterns from spatial datasets [1-3]. Extracting interesting patterns from spatial datasets is more difficult than extracting the corresponding patterns from transaction datasets due to the complexity of spatial data types, spatial relationships and spatial autocorrelation [4]. A spatial co-location pattern represents a subset of spatial features whose instances are frequently located in a spatial neighborhood. For example, botanists have found that there are orchids in 80% of the area where the middle-wetness green-broad-leaf forest grows. Spatial co-location patterns may yield important insights for many applications. For example, a mobile service provider may be interested in mobile service patterns frequently requested by geographical neighboring users. The locations which are gotten together by people can be used for providing attractive location-sensitive advertisements, etc. Other application domains include Earth science, public health, biology, transportation, etc.

Co-location pattern discovery presents challenges due to the following reasons: First, it is difficult to find co-location patterns with traditional association rule mining algorithms since there is no concept of traditional "transaction" in most of spatial datasets [1,5,6]. Second, the instances of a spatial feature distribute in spatial framework and share complex spatial neigh borhood relationships with other spatial instances. So a large fraction of the computation time of mining co-location patterns is devoted to generating the table instances of co-location pattern.

In this paper, a novel approach for mining co-location patterns is proposed. This method keeps the *Apriori*-like approach to generate size-k prevalence co-locations after size-(k-1) prevalence co-locations. Considering efficient generating co-location instances, an improved co-location pattern instance tree (called iCPI-tree) is defined. Then an iCPI-tree based co-location pattern mining algorithm is designed. The time and space complexity of the algorithm are analyzed. The experimental evaluations using synthetic and real world datasets show the iCPI-tree algorithm outperforms the other algorithms which will be mentioned in this paper and is scalable in dense spatial datasets.

The reminder of the paper is organized as follows. Section 2 gives an overview of the basic concepts of co-location pattern mining and the problem definition, and then discusses related works and motivation. The iCPI-tree approach is introduced in Section 3. Section 4 presents the proofs of completeness and correctness of the new algorithm, and gives computational efficiency analysis of the algorithm. The experimental results are presented in Section 5. Section 6 summarizes our study and points out some future research issues.

2. CO-LOCATION PATTERN MINING

2.1. Basic Concepts

Given a set of spatial features *F*, a set of their instances *S*, and a spatial neighbor relationship *R* over *S*. *R* could be topological relationships (e.g. linked, intersection), distance relationships (e.g. Euclidean distance metric) and mixed relationships (e.g. the shortest distance of two points on a map).

*Address correspondence to this author at the Department of Computer Science and Engineering, School of Information, Yunnan University, Kunming, 650091, P. R. China; E-mail: lzhwang@ynu.edu.cn

1874-1339/09

As shown in Fig. (**1**), there are 4 spatial features *A*, *B*, *C* and *D* and their instances. *A*.1 stands for the first instance of feature *A*. If *R* is defined as a Euclidean distance metric and its threshold value is *d*, two spatial objects are neighbors if they satisfy the neighbor relationship:

$$R(\text{A.1},\text{B.1}) \Leftrightarrow (distance(\text{A.1},\text{B.1}) \le d)$$

Given a subset of spatial instances $I = \{i_1, i_2, ..., i_m\}$, $I \subseteq S$. I is called as an **R-neighbor** if *I* forms a clique under the neighbor relation *R*.

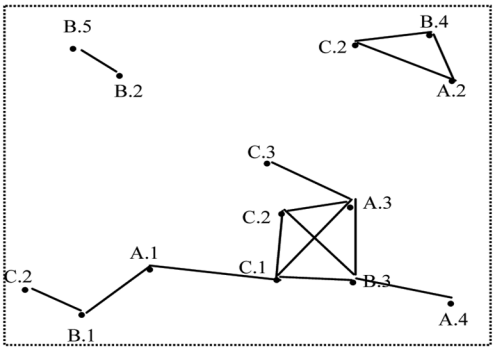

Fig. (1). An example of spatial feature instances.

A **co-location** *c* is a subset of spatial features, i.e., $c \subseteq F$. An R-neighbor *I* is a **row instance** of a co-location *c* if *I* contains instances of all the features in *c* and no proper subset of it does so. The **table instance** of a co-location *c* is the collection of all row instances of *c*. The **size** of a co-location *c* is the number of spatial features in co-location *c*, it is denoted as $size(c) = |c|$.

The interest degree of a co-location differs from the degree of support in traditional association rules mining. A new prevalence measure concept called the **participation index** is introduced by Huang, Shekhar and Xiong in [7]. Participation ratio will be presented before giving the concept of participation index.

The **participation ratio** $PR(c, f_i)$ for feature type f_i in a size-k co-location $c = \{f_1 ... f_k\}$ is the fraction of instances of feature f_i which participate in any row instance of co-location *c*. The participation ratio can be computed as $PR(c, f_i) = \frac{|\pi_{f_i}(table\_instance(c))|}{|table\_instance(f_i)|}$, where π is the relational projection operation with duplication elimination.

The **participation index** of a co-location $c = \{f_1 ... f_k\}$ is the minimum in all $PR(c, f_i)$ of co-location *c*:

$$PI(c) = \min_{i=1}^{k} \{PR(c, f_i)\}.$$

Example 1 Take Fig. (**1**) as an example. *A* has 4 instances, *B* has 5 instances, and *C* has 3 instances. Suppose co-location $c = \{\text{A,B,C}\}$, the table instance of co-location *c* has {{A.2,B.4,C.2}, {A.3,B.3,C.1}, {A.3,B.3,C.2}}. $PR(c, \text{A}) = 2/4$ since there are only *A.2* and *A.3* in the table instance. Similarly, $PR(c, \text{B}) = 2/5$, $PR(c, \text{C}) = 2/3$. $PI(C) = \min(PR(c, \text{A}), PR(c, \text{B}), PR(c, \text{C})) = 2/5$.

Given a minimum prevalence threshold *min_prev*, a co-location *c* is a **prevalent co-location** if $PI(c) \ge min\_prev$ holds.

Lemma 1 The participation ratio and the participation index are monotonically non-increasing with the size of the co-location increasing.

Proof Suppose a spatial instance is included in the table instances of co-location *c*. For co-location $c' \subseteq c$, the spatial instance *e* must be included in the table instances of c'. The opposite is not true. Therefore, the participation ratio is monotonically non-increasing.

Suppose $c = \{e_1, \cdots, e_k\}$,

$$PI(c \cup e_{k+1}) = \min_{i=1}^{k+1} \{PR(c \cup e_{k+1}, e_i)\}$$

$$\le \min_{i=1}^{k} \{PR(c \cup e_{k+1}, e_i)\} \le \min_{i=1}^{k} \{PR(c, e_i)\} = PI(c)$$

Therefore, the participation index of co-location is also monotonically non-increasing.

Lemma 1 ensures that the participation index can be used to effectively prune the search space of co-location pattern mining.

2.2. Problem Definition

The co-location mining problem is formalized as follows. We focus on finding a correct and complete set of co-location rules with reducing the computation cost.

Given:

1) A spatial framework η
2) A set of spatial features $F = \{f_1, ..., f_n\}$ and a set of their instances $S = S_1 \cup S_2 \cup ... \cup S_n$, where $S_i (1 < i < n)$ is the set of instances of the feature f_i and each instance is a vector <feature type, instance id, location>, where location $\in \eta$.
3) A spatial neighbor relation *R* over *S*.
4) A minimum prevalence threshold *min_prev* and a minimum conditional probability threshold *min_cond_prob*.

Find:

A set of co-location rules with participation index $\ge$ *min_prev* and conditional probability $\ge$ *min_cond_prob*.

Objective:

1) Find a correct and complete set of co-location rules.
2) Reduce the computation cost.

Constraints:

1) R is a distance metric based neighbor relationship and has symmetric property.

2) The spatial dataset is a point dataset.

2.3. Related Work

In previous work on mining co-location patterns, Morimoto [4] defined distance-based patterns called k-neighboring class sets. In his work, the number of instances for each pattern is used as the prevalence measure, which does not possess an anti-monotone property by nature. However, Morimoto used a non overlapping instance constraint to get the anti-monotone property for this measure. In contrast, Shekhar & Huang [8] developed a feature centric model, which does away with the non-overlapping instance constraint. The related works in the approach proposed by Shekhar & Huang can be classified into three kinds for identifying co-location table instances: the full-join approach [7], the partial-join approach [9] and the join-less approach [10,11].

The *full-join* approach is mainly based on the computation of the join operation between table instances for identifying co-location instances. This approach is similar to *Apriori* method and it could generate correct and complete prevalent co-location sets. However, scaling the algorithm to substantially large dense spatial datasets is challenging due to the increasing number of co-locations and their table instances.

The *partial-join* approach is to build a set of *disjoint clique* in spatial instances to identify the intraX instances of co-location (belonging to a common clique) and interX instances of co-location (all instances have at least one cut neighbor relation), and merge the intraX instances and interX instances to calculate the value of the *PI*. This approach reduces the number of expensive join operations dramatically in finding table instances. However, the performance depends on the distribution of the spatial dataset, exactly the number of cut neighbor relations.

The *join-less* approach puts the spatial neighbor relationships between instances into a compressed *star neighborhood*. All the possible table instances for every co-location pattern were generated by scanning the star neighborhood, and by 3-time filtering operation. The join-less co-location mining algorithm is efficient since it uses an instance-lookup scheme instead of an expensive spatial or instance join operation for identifying co-location table instances. So the idea of the join-less is great. However, the star neighborhood structure is not an ideal structure for generating table instances, for the table instances generating from this structure have to be filtered. Therefore, the computation time of generating co-location table instances will increase with the growing of length of co-location patterns.

The *CPI-tree* algorithm proposed by Wang *et al.* in [11] is a new join-less algorithm. In this method, a new structure called *CPI-tree* (Co-location Pattern Instance Tree) is introduced. It could materialize the neighbor relationships of a spatial data set, and find all the table instances recursively from it. Different from the star neighborhood structure in the join-less approach of the paper [10], all information of the neighbor relationships in a spatial dataset is organized together by the CPI-tree. So, the third phase filter in the join-less algorithm, which might be an expensive step, need not be performed. However, this method gives up the *Apriori*-like model, i.e., to generate size-k prevalence co-locations after size-(k-1) prevalence co-locations. In many cases the *Apriori* candidate generate-test method reduces the number of candidate sets significantly and leads to performance gain.

Besides the above representative co-location mining algorithms, Huang, Pei and Xiong address the problem of mining co-location patterns with rare spatial events [12]. In this paper, a new measure called the maximal participation ratio (maxPR) was introduced and a week monotonicity property of the maxPR measure was identified. Verhein and Al-Naymat considered mining complex spatial co-location patterns from spatial dataset [13]. They introduced the idea of maximal clique and applied the GLIMIT [14] (it is a very fast and efficient itemset mining algorithm that has been shown to outperform Apriori and FP-Growth) itemset mining algorithm to their task. Celik *et al.* studied zonal co-location patterns discovery problem [15].

2.4. Motivation

Let us see the spatial instances in Fig. (**1**). If a pair of spatial instances satisfy neighbor relationship R, connect them with a solid line (as shown in Fig. **1**), then a graph *G* can be obtained. Each co-location instance is a complete subgraph (clique) in *G*. Mining co-location patterns is equal to the process of mining all cliques in *G* and calculating the *PI* value of each co-location pattern. However, such process has been proved as a *NP-Hard* problem [16]. In fact, in the process of finding cliques, according to the definition of co-location pattern, the same spatial features cannot appear in a clique, and according to the anti-monotonic property of *PI* value (**Lemma 1**), not all the cliques should be calculated. The most existed co-location pattern mining algorithms adopt an *Apriori*-like approach.

Can the cliques be calculated efficiently by simply scanning *G*? Can a structure which contains the information of table instances be built? In this paper, a new structure called iCPI-tree (improved Co-location Pattern Instance Tree), which is an improved version of the CPI-tree [11], will be introduced. The join-less idea used in the join-less approach [10] and CPI-tree approach [11] is efficient since it uses an instance-lookup scheme instead of an expensive spatial or instance join operation for identifying co-location table instances, but their efficiency depends on the distribution of the datasets. Because for join-less approach, the third phase filter might be an expensive step when there are many non-co-location instances in the star instances of candidate co-locations. For CPI-tree approach, though the table instances generating is efficient, the storing and searching of whole CPI-tree will be a problem with input datasets become bigger. In addition, no candidate pruning by **Lemma** 1 will bring on mining algorithms require time-consuming in the

dense datasets. Therefore, remaining the *Apriori* candidate generate-test model of the join-less approach and the tree structure advantage of the CPI-tree approach, a new iCPI-tree method is proposed in this paper. The new method is expected to reduce the computation cost of co-location pattern mining in any kinds of datasets.

3. AN ICPI-TREE BASED APPROACH FOR CO-LOCATION PATTERN MINING

In this section, an iCPI-tree based approach for mining co-location patterns is discussed. First, an iCPI-tree data structure is defined, and then the iCPI-tree based co-location mining algorithm is presented.

3.1. iCPI-tree

Although the CPI-tree approach [11] materializes the neighbor relationships of a spatial dataset for efficient co-location instance identifying, the storages of the CPI-tree and co-location instances are expensive. In addition, the number of recursions in the CPI-tree algorithm is also enormous with the size of datasets become huge. In other words, the performance of the CPI-tree algorithm should be improved. An improved CPI-tree is proposed for more efficient co-location instance identifying.

Definition 1 Given a subset of spatial instances $I=\{i_l,...,i_v\}$, $l,v\in\{1,2,\cdots,m\}$, If $i_i \le i_j$ (the spatial features in alphabetic order, and then the different instance of the same spatial feature in numerical order) holds for any $l \le i \le j \le v$, the I is called as an **ordered instance set**. If I is a table instance, it is called as an **ordered table instance**. If the feature-name of i_i is not the same as the feature-name of i_l and $R(i_l,i_i)$(represents i_l and i_i is neighbor) holds for any $l < i \le v$, The I is called as **ordered neighbor relationship set of the instance** i_l. The set of ordered neighbor relationship sets of all instances of a spatial feature *x* is denoted as δ_x.

Example 1 Take Fig. (**1**) as an example. Spatial feature *A* has 4 instances, *B* has 5 instances, and *C* has 3 instances. Two instances are connected if they are neighbors in Fig. (**1**). Therefore, *I={A.3,B.3,C.1}* is an ordered instance set, it is also an ordered table instance. The ordered neighbor relationship set of the instance *A.3* is {*A.3,B.3,C.1,C.2,C.3*}. The set of ordered neighbor relationship sets of all instances of the feature *A* is denoted as δ_A**={**{*A.1,B.1,C.1*}, {*A.2,B.4,C.2*}, {*A.3,B.3,C.1,C.2,C.3*}, {*A.4,B.3*}}.

The concept of iCPI-tree can be defined based on the following observations:

(1). Since spatial neighbor relationships between instances make certain all table instances, it is necessary to perform one scan of spatial datasets to identify the set of spatial neighbor relationships.

(2). If the set of neighbor relationships can be stored in an ideal data structure, it may be possible to avoid repeatedly scanning the set of neighbor relationships. Furthermore, it may be cost-efficient to generate table instances using a pile-instance-lookup scheme instead of instance join operation.

(3). The recursive and hierarchical properties of tree structure ensure the clarity and simplicity of the algorithms' description. If all spatial instances are sorted in ascending order (the spatial features in alphabetic order, and then the different instance of the same spatial feature in numerical order), a graph G representing spatial neighbor relationships may correspond to a unique tree structure.

With the above observations, a tree structure (called iCPI-tree (Improved Co-location Pattern Instance Tree), for all table instances can be generated in batch from it) can be defined as follows.

Definition 2 (iCPI-tree). Given a set of spatial features $F=\{f_1,...,f_n\}$ and a set of ordered instance neighbor relationship of spatial features $\delta=\delta_{f_1}\cup\delta_{f_2}\cup...\cup\delta_{f_n}$, $\delta_{f_i}(1<i<n)$ is the set of ordered neighbor relationship sets of all instances of the feature f_i, a tree designed as below is called as an **improved co-location pattern instances tree** (*iCPI-tree*, for short).

1). It consists of one root labeled as "*null*", a set of the spatial feature sub-trees as the children of the root.

2). The spatial feature f_i sub-tree consists of the root f_i and each subset of δ_{f_i} as a branch of the root. Each branch records an ordered neighbor relationship set of corresponding instance and relevant feature-name.

Example 2 Fig. (**2**) is the *iCPI-tree* of the example in Fig. (**1**). The feature '*A*' sub-tree consists of the root '*A*' and branches *A.1, A.2, A.3,* and *A.4*. The branch *A.1* records the content of ordered neighbor relationship set of the instance *A.1* and relevant feature-name, i.e., there are $R(A.1,B.1)$ and $R(A.1,C.1)$.

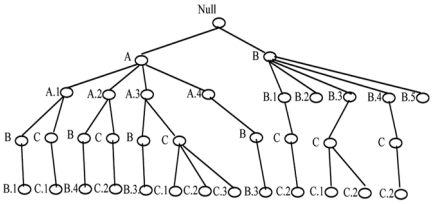

Fig. (2). The iCPI-tree of the example in Fig. (**1**).

The iCPI-tree of a spatial dataset constructed based on definition 2 will be unique. The iCPI-tree materializes the neighbor relationships of a spatial dataset with no duplication of the neighbor relationships and no loss of co-location instances, and the more important thing is that it is convenient and efficient to generate the co-location instances from it.

3.2. iCPI-tree Based Co-Location Mining Algorithm

The iCPI-tree based co-location mining algorithm has two phases. The first phase converts an input spatial dataset into an iCPI-tree for efficient co-location instance identifying. The second phase recursively generates prevalent co-locations and co-location rules based on the iCPI-tree. In this phase, Size-k table instances are expanded from the size-(k-1) table instances whose object feature types are the same as the first (k-1) features of the candidate co-locations. Fig. (**3**) illustrates an iCPI-tree based algorithm trace. Algorithm 1 shows the pseudo code.

Algorithm 1 iCPI-tree based co-location mining algorithm.

Input

$F = \{f_1,..., f_n\}$: a set of spatial feature types;

S: a set of spatial instances and each instance is a vector <feature type, instance id, location>;

R: the spatial neighbor relationship (e.g. Euclidean distance);

min_prev: prevalence value threshold;

min_cond_prob: conditional probability threshold;

Output

A set of all prevalent co-location rules with participant index greater than *min_prev* and conditional probability greater than *min_cond_prob*;

Variables

K: co-location size;

δ : A set of spatial ordered neighbor relationships between instances;

C_k: a set of size-k candidate co-locations;

P_k: a set of size-k prevalent co-locations;

I_k: a set of table instances of co-locations in C_k;

Method

1) δ = gen_neighborhood (F, S, R);
2) iCPI-tree= gen_iCPI-tree (δ , F);
3) P_1=F; I_1=S; K=2;
4) **While** (not empty P_{k-1}) **Do**
5) C_k=gen_candidate_colocations(P_{k-1});
6) I_k=gen_instances(I_{k-1},*ICPI-tree*,C_k);
7) P_k=gen_prev_co-location (C_k,I_k,*min_prev*);
8) R_k=gen_co-location_rule (P_k,I_k,*min_conf*);
9) K=K+1
10) Enddo
11) Return U(R_2,…, R_k)

Convert a Spatial Dataset to a Set of Spatial Ordered Neighbor Relationships Between Instances (Step 1): Given an input dataset and a neighbor relationship, first find all neighboring object pairs using a geometric method such as plane sweep [17], or a spatial query method using quaternary tree or R-tree [18]. The set of ordered neighbor relationships are generated by grouping the neighborhoods sorted by the feature type in lexical order. The set of spatial ordered

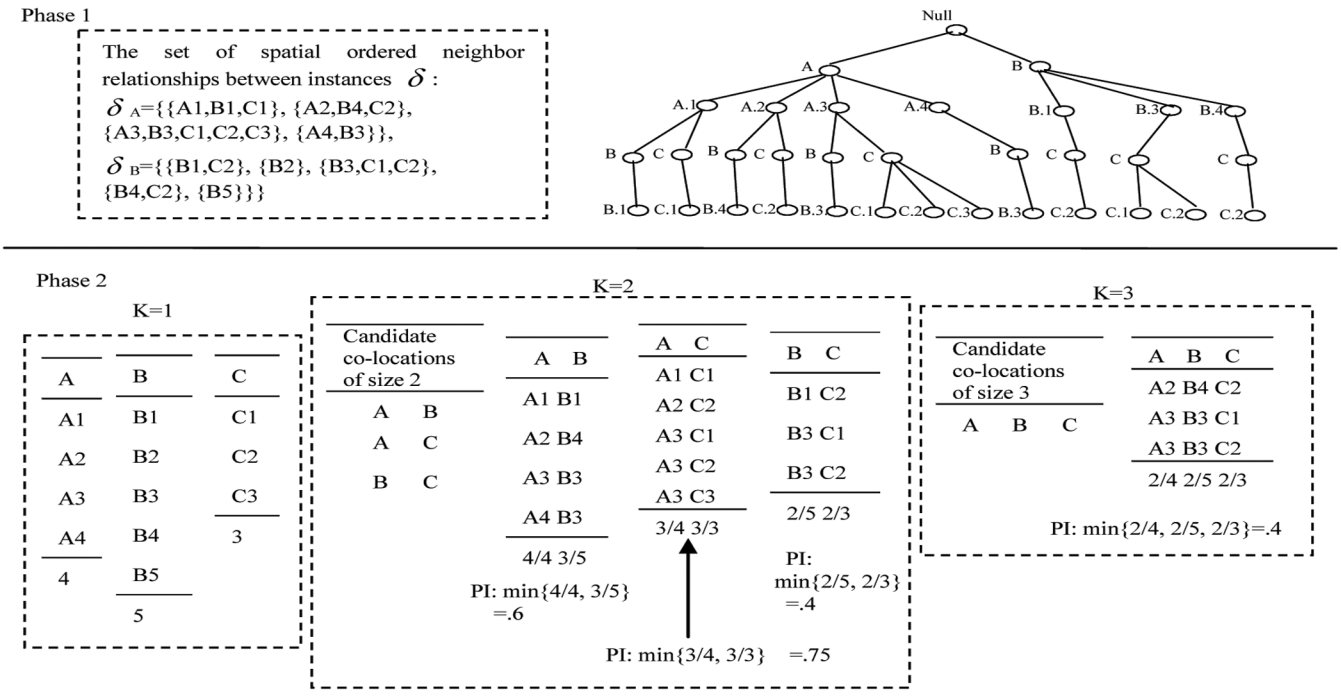

Fig. (3). iCPI-tree algorithm trace.

neighbor relationships between instances is denoted as δ. Fig. (**3**) shows the ordered neighborhoods sorted by the feature type.

Generate the iCPI-tree of the Set of Spatial Ordered Neighbor Relationships (Step 2): From the set of ordered spatial neighbor relationships between spatial instances δ and a set of spatial features $F = \{f_1,..., f_n\}$, the iCPI-tree can be built by iteratively creating *n-1* spatial features branches of the iCPI-tree. Each branch is created by scanning a sub-set of the set δ. For example, in Fig. (**3**), scanning the subset δ_A={*{A1,B1,C1}, {A2,B4,C2}, {A3,B3,C1,C2, C3}, {A4,B3}*}, the branch *A* of the iCPI-tree is built. This step is specified as follows.

Procedure Gen_iCPT-tree (δ, *F*)

Input

F: A set of spatial features.

$\delta = \left\{\delta_{f_1} = \{\delta_{f_1}^{l_1},...\delta_{f_1}^{l_{k_1}}\}, \delta_{f_2} = \{\delta_{f_2}^{l_1},...\delta_{f_2}^{l_{k_2}}\}...\delta_{f_n} = \{\delta_{f_m}^{l_1},...\delta_{f_m}^{l_{k_m}}\}\right\}$: A set of spatial ordered neighbor relationships between instances, where $\delta_{f_i}\ (1 \le i \le n)$s the set of the set $\delta_{f_i}^{l_i}$ of ordered neighbor instances (they are "bigger" than the instance l_i) of instances l_i of feature f_i, whose order is sorted in ascending order.

Output

iCPI-tree: An improved co-location pattern instance tree.

Method

1) Create a root "Null" for iCPI-tree;
2) i=1;
3) **While** i<n **Do**;
4) **{** Create a sub-tree f_i of the root "Null";
5) Create a branch $\delta_{f_i}^{l_1}$ for sub-tree f_i;
6) **For** each $\delta_{f_i}^{l_i}\ (1 < i \le k_i)$ of δ_{f_i} in δ
7) create a child-node of the branch $\delta_{f_i}^{l_1}$;
8) i=i+1;
9) **}**
10) **Return** the root '*Null*'

Assign Starting Values to Various Data Structures Used in the Algorithm (Step 3): First, all features to size-1 prevalent co-locations by the definition of the participation index measure. The number of instances per feature can be known during the scan of the input spatial dataset for computing the set of ordered neighbor relationships.

Iteratively Mining Co-Location Rules (Step 4-10): Step 4) to Step 10) of algorithm iteratively perform four basic tasks, namely, generation of candidate co-locations, generation of table instances of candidate co-locations, pruning, and generation of co-location rules. These tasks are carried out a loop iterating over the size of the co-locations. Iterations start with size-2 since the definition of prevalence measure allows no pruning for co-locations of size-1.

3.2.1. Generation of Candidate Co-Locations

Size-k (k>1) candidate co-locations are generated from prevalent size-(k-1) co-locations. Here, we have a feature level pruning of candidate co-locations. If any subset of a candidate co-location is not prevalent, the candidate co-location is pruned.

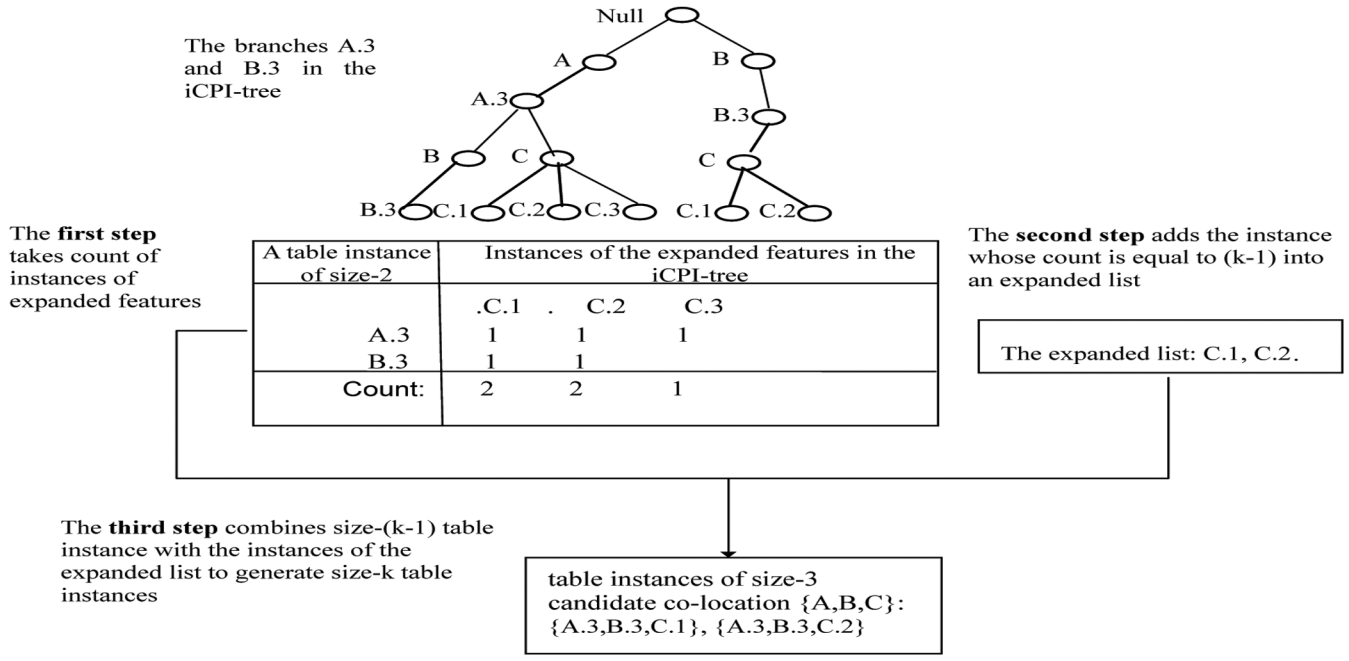

Fig. (4). The process of generating table instances based on the iCPI-tree.

3.2.2. Generation of Table Instances of Candidate Co-Locations

Size-k table instances are expanded from the size-(k-1) table instances whose object feature types are the same as the first (k-1) features of the candidate co-locations. The operation of expanding is performed based on the iCPI-tree. For example, the table instances of candidate co-location {*A,B,C*} are expanded from the table instances of co-location {*A,B*}. The expanded process has three steps. The first step takes count of instances of expanded features based on the iCPI-tree. The second step adds the instance whose count is equal to (k-1) into an expanded list. The third step combines size-(k-1) table instance with the instances of the expanded list to generate size-k table instances. Fig. (**4**) shows the process of generating table instances of candidate co-location {*A,B,C*} from table instance {A.3,B.3} of size-2 co-location {*A,B*}. As shown in Fig. (**4**), In the iCPI-tree, instances of feature "*C*" can be expanded from instance "*A.3*" are "*C.1*", "*C.2*" and "*C.3*", while child of "*B.3*", whose feature is "*C*", is "*C.1*" and "*C.2*". They are counted in the first step. Then instance "*C.1*" and "*C.2*" are added into expanded list since its count is 2 (i.e., k-1). Finally, the table instance {*A.3, B.3*} is combined with "*C.1*" and "*C.2*" of expanded list to generate two table instances {*A.3,B.3,C.1*} and {*A.3,B.3,C.2*} of candidate co-location {*A,B,C*}.

3.2.3. Pruning

Candidate co-locations can be pruned using the given threshold *min_prev* on prevalence measure. In addition, iCPI-tree pruning can be used for more efficient identifying table instances from the iCPI-tree.

Prevalence-Based Pruning

The prevalence-based pruning of co-locations is done by the participation index values calculated from the set of co-location instances I_k. Bitmaps data structure can be used for efficient computation the participation index of a candidate co-location [7]. Prevalent co-locations satisfying the threshold *min-prev* are selected. For each selecting prevalent co-location *c* after prevalence-based pruning, a counter to specify the cardinality of the table instance of *c*. The relevant (since the generation of the table instances of candidate co-location {A,B,C} uses only the table instances of co-location {A,B} based on iCPI-tree) table instances of the prevalent co-locations in this iteration will be kept for generation of the prevalent co-locations of size-(k+1) and discarded after the next iteration.

iCPI-tree-Based Pruning

Although generating co-location instances from a iCPI-tree will be No loss of co-location instances and no duplication of co-location instances, the following pruning strategies can be used to improve efficiency of generating co-location instances from CPI-tree.

Pruning 1

A node, which is the child of the branch "f_i" ($1 \leqslant i \leqslant n$) and has no child, can be pruned.

Proof. If a node is the child of the branch "f_i" ($1 \leqslant i \leqslant n$) and it has not a child node, it must be the spatial instance without neighborhood. So it can be pruned.

Example 3 In Fig. (**2**), the nodes B.2 and B.5 can be pruned from the iCPI-tree with **Pruning 1**.

Pruning 2

By using **Pruning 1**, If the number of the pruned instances of a feature f_i is greater than *min_prev**|f_i|, then all the instance nodes of the feature f_i and, the relevant edges and the child nodes in the iCPI-tree can be pruned.

Proof. If the number of the pruned spatial instances of a feature f_i with **Pruning 1** is greater than *min_prev**| f_i|, the number of the remaining instances of the feature is less than the *min_prev**| f_i|. Therefore, all instances of this spatial feature might be pruned due to the co-location containing the feature might not be prevalent.

Example 4 Suppose that three instances of spatial feature B was pruned with **Pruning 1**, and there are five instances in feature B and the *min_prev* is 50%, then all the instances of B and, the relevant edges and child nodes can be pruned with **Pruning 2**.

3.2.4 Generating Co-Location Rules

The gen_co-location_rule function generates all the co-location rules satisfying the user defined threshold *min_cond_prob* from the set of prevalent co-locations and their table instances. Bitmaps or other data structures can be used for efficient computation using the same strategies for prevalence-based pruning.

4. ANALYSIS OF THE ICPI-TREE ALGORITHM

Here, the iCPI-tree based co-location mining algorithm for completeness, correctness and computational complexity is analyzed.

4.1. Completeness and Correctness

Completeness means the iCPI-tree algorithm finds all co-location rules whose participation index and conditional probability satisfy a user specified minimum prevalence threshold *min_prev* and conditional probability threshold *min_cond_prob*. Correctness means that all co-location rules generated by the iCPI-tree algorithm have a participation index and a conditional probability above the *min_prev* and *min_cond_prob*. First related lemmas are provided.

Lemma 2 The iCPI-tree model does not miss any neighbor relationships of an input spatial data.

Proof: according to Definition 2, all the spatial instances are scanned and their neighbor relationships are recorded in an iCPI-tree. Therefore, none of the spatial instance neighbor relationships is missed in CPI-tree.

Lemma 3 The iCPI-tree materializes the neighbor relationships of an input spatial data with no duplication of the spatial neighbor relationships.

Proof: according to Definition 1, there are not duplication spatial neighbor relationships in the set of ordered neighbor relationships. So, it is obvious because each branch of an iCPI-tree records an ordered neighbor relationship set of corresponding instance.

Lemma 4 Given a size-(k-1) table instance $I_{k-1}=\{o_1,\ldots,o_{k-1}\}$ of a co-location $C_{k-1}=\{f_1,\ldots,f_{k-1}\}$. If any spatial instance o_k of a spatial feature f_k (f_k is not belong to C_{k-1}) is a child-node of each instance o_i ($1 \leqslant i \leqslant k-1$), the co-location instance $I_k=\{o_1,\ldots o_{k-1},o_k\}$ is a table instance of co-location $C_k=\{f_1,\ldots,f_{k-1},f_k\}$.

Proof: First the size-(k-1) table instance $I_{k-1}=\{o_1,\ldots,o_{k-1}\}$ means each instance o_i ($1 \leqslant i \leqslant k-1$) has neighbor relationships to all other instances in I_{k-1}. Second if the instance o_k of a spatial feature f_k (f_k is not belong to C_{k-1}) is a child-node of the nodes $o_1,\ldots,o_{k-1}$, the instance o_k has neighbor relationship to the instance o_i ($1 \leqslant i \leqslant k-1$). Thus each instance o_i ($1 \leqslant i \leqslant k$) has neighbor relationships to all other instances in I_k since the neighbor relationship is symmetric. The co-location instance $I_k=\{o_1,\ldots,o_k\}$ is a table instance of co-location $C_k=\{f_1,\ldots,f_k\}$.

Theorem 1 The iCPI-tree based co-location mining algorithm is complete.

Proof: The completeness of the iCPI-tree algorithm can be shown by the following two parts. The first is that the method to materialize the neighbor relationships of an input spatial data based on the iCPI-tree (step 1 and step 2) is correct. The iCPI-tree does not miss and duplicate any neighbor relationship of an input spatial data by **Lemma 2** and **Lemma 3**. The method to generate co-location instances from iCPI-tree is correct by **Lemma 4**. Next, it is shown that no table instance can be generated out of the method. Suppose a size-k table instance can be generated out of the method. If this is a size-2 table instance, and then there is not a parent-child-link between the two instances in iCPI-tree. According to **lemma 2**, there is not a spatial neighbor relationship between the two instances, this reduces to absurdity. For size-k table instances $I_k=\{o_1,\ldots o_{k-1},o_k\}$ ($k>2$), the instance-node o_k is at least not a child-node of a instance-node o_i ($1 \leqq i \leqq k-1$) in the iCPI-tree. According to **lemma 2**, there is not a spatial neighbor relationship between the two instances. This also reduces to absurdity.

Theorem 2 The iCPI-tree co-location mining algorithm is correct.

Proof: The correctness of the iCPI-tree algorithm can be guaranteed by step 7 and 8. Step 7 selects only co-locations whose participation index satisfies a user specific prevalence threshold *min_prev*. The generated co-location rules by step 8 also satisfy a user specific conditional probability *min_cond_prob*.

4.2. Computational Complexity Analysis

This section analyzes the time and space complexity of the new method and then, compares the computational cost of the iCPI-tree algorithm with the join-based algorithm, the join-less algorithm and the CPI-tree algorithm.

Time complexity: The time complexity of the **algorithm** includes Gen_neighborhood, gen_iCPI-tree, and the loop step 4-10. Suppose m is the total number of instances of all features. In the worse case, the computational complexity of the procedure *Gen_neighborhood* will be $O(m^2 \log_2 m)$. For procedure *gen_iCPI-tree*, if N_{ins} is the number of spatial neighbor relationships, the cost is $O(N_{ins}) \leq O(m^2)$.

For the loop k of Step 4 in **Algorithm 1**, the bulk cost is to generate co-location instances I_k of the set of candidate co-locations C_k. This cost depends on the number of spatial instances, the number of features, the number of spatial neighbor relationships between instances, the number and the size of candidate co-locations, and the number of table instances in co-locations. But by sorting spatial instances and co-locations, and using expanding method based on the iCPI-tree to generate table instances, which dramatically reduces the cost of algorithms. The real performance of the algorithm is discussed in Section 5.

Computational cost comparison: Let T_{icpi}, T_{cpi}, T_{jb} and T_{jl} represent the costs of the iCPI-tree algorithm, the CPI-tree algorithm, the join-based algorithm and the join-less algorithm respectively.

$$\begin{aligned} T_{iCPI} &= T_{gen\_neib\_\delta}(dataset) + T_{gen\_iCPI-tree}(\delta) + \\ &\sum_k (T_{gen\_candi}(P_{k-1}) + T_{gen\_inst}(I_{k-1}, iCPI-tree) \\ &+ T_{prune}(C_k)) \\ &\approx \sum_k T_{gen\_inst}(I_{k-1}, iCPI-tree) \end{aligned} \quad (1)$$

$$\begin{aligned} T_{cpi} &= T_{gen\_neib\_\delta}(dataset) + T_{gen\_CPI-tree}(\delta) + \\ &T_{gen\_inst}(Ins_{k-1}) + T_{prune}(C_k)) \\ &\approx T_{gen\_inst}(CPI-tree) \end{aligned} \quad (2)$$

$$\begin{aligned} T_{jb} &= T_{gen\_size-2\_col}(dataset) + \sum_k (T_{gen\_candi}(P_{k-1}) + \\ &T_{gen\_inst}(CPI-tree) + T_{prune}(C) \\ &\approx \sum_k T_{gen\_inst}(Ins_{k-1}) \end{aligned} \quad (3)$$

$$\begin{aligned} T_{jl} &= T_{gen\_star-neib}(dataset) + \sum_k (T_{gen\_candi}(P_{k-1}) + \\ &T_{gen\_star\_inst}(C_k, star-neib) + \\ &T_{filter\_star\_inst}(C_k, star-neib) + T_{prune}(C_k)) \\ &\approx \sum_k (T_{gen\_star\_inst}(C_k, star-neib) + \\ &T_{filter\_clique\_inst}(C_k, star-neib)) \end{aligned} \quad (4)$$

In the above Equation (1), $T_{gen\_neib\_\delta}(dataset)$ represents the cost of generating ordered neighbor relationship set

δ with the dataset. $T_{gen\_iCPI-tree}(\delta)$ represents the cost of building the iCPI-tree. $\sum_k T_{gen\_candi}(P_{k-1})$ is the cost of generating all candidate co-locations. $\sum_k T_{gen\_inst}(I_{k-1}, iCPI-tree)$ represents the cost of identifying table instances of all candidate co-locations based on the iCPI-tree and relevant size-(k-1) co-location instances. $\sum_k T_{prune}(C_{k+1})$ is the cost for pruning non-prevalent size *k* co-locations. The bulk of time is consumed in identifying table instances of all candidate co-locations.

In equation (2), $T_{gen\_neib\_\delta}(dataset)$ represents the cost of generating ordered neighbor relationship set δ with the dataset. $T_{gen\_CPI-tree}(\delta)$ is the cost of building the CPI-tree based the δ. $T_{gen\_inst}(CPI-tree)$ represents the cost of generating all co-location instances from CPI-tree. $T_{prune}(C)$ is the cost for pruning non-prevalent co-locations. The bulk of time is consumed in generating all co-location instances from CPI-tree.

In equation (3), $T_{gen\_size-2\_col}(dataset)$ represents the cost of generating size-2 co-locations and their table instances with the dataset. $\sum_k (T_{gen\_candi}(P_{k-1}) + T_{gen\_inst}(Ins_{k-1}) + T_{prune}(C_k))$ represents the all cost of generating all prevalence co-locations, where $T_{gen\_candi}(P_{k-1})$ is the cost of generating size *k* candidate co-location with the prevalent size *k-1* co-locations, $T_{gen\_inst}(Ins_{k-1})$ represents the cost of generating table instances of size *k* candidate co-locations with size *k-1* table instances, $T_{prune}(C_k)$ is the cost for pruning size *k* co-locations. The bulk of time is consumed in generating table instances of all candidate co-locations.

In equation (4), $T_{gen\_star-neib}(dataset)$ represents the cost of converting a spatial dataset to a disjoint star neighborhood. $\sum_k T_{gen\_candi}(P_{k-1})$ is the cost of generating all candidate co-locations. $\sum_k (T_{gen\_star\_inst}(C_k, star-neib) + T_{filter\_clique\_inst}(C_k, star-neib))$ is the cost of generating the star instances and filtering co-location instances of all candidate co-locations with the star neighborhoods. $T_{prune}(C_k)$ is the cost for pruning non-prevalent size *k+1* co-locations. The bulk of time is consumed in generating the star instances and filtering co-location instances of size *k* candidate co-locations with the star neighborhoods.

The difference of the three algorithms' computational cost is affected by the number of table instances, the number of candidate co-locations and the distribution of spatial features and spatial instances. When the number of table instances and candidate co-locations increase, the cost of the join-based algorithm, the join-less algorithm and the CPI-tree algorithm are greater than the cost of the iCPI-tree algorithm. This happens because the pile-instance-lookup scheme (as shown in Fig. **4**) based on the iCPI-tree improves the running performance of identifying table instances in the iCPI-tree algorithm. In our experiments, as described in the next section, we use the data density, the prevalence threshold *Min_prev* and the neighbor distance *d* as key parameters to evaluate the algorithms. We can expect that the iCPI-tree approach is likely more efficient than the join-based method, the join-less method and the CPI-tree when the spatial dataset is dense (containing many table instances).

Space Complexity

The store space of the tree iCPI-tree is the most costly in the algorithm, if it is always in the main memory, the space cost of the algorithm is $O(N_{ins}) \le O(m^2)$. But a method which partial sub-trees of the iCPI-tree are remained to reduce the need of the space can be adopted, because in one iterative of generating a candidate co-location *c*, the instances of features related to the *c* only need to be in the main memory.

5. EXPERIMENTAL RESULTS

In this section, the performance of the algorithms is evaluated with the join-based approach, the join-less approach and the CPI-tree approach using both synthetic and real datasets. All the experiments were performed on a 3-GHz Pentium PC machine with 2G megabytes main memory, running on Microsoft Windows/XP. All programs are written in Java.

Synthetic datasets are generated using a methodology similar to the methodology used in paper [7], which has 20 spatial features and 11292 spatial instances in an area 1000×1000. The synthetic data generator allows us to better control the study of the algorithms and the effects of interesting parameters.

To test the practicability of the iCPI-tree method, a real dataset, the plant distribution dataset of the "Three Parallel Rivers of Yunnan Protected Areas" area, is used. It contains the number of plant species (feature types) is 16. The total number of plant instances is 3908. When *Min_prev* and distance *d* are set to 0.1 and 1900 respectively, the maximum size of co-location is 4 and the total number of size 2 co-location patterns is 42. There are a huge number of spatial neighbor relationships between instances due to the plants' particularity of growing in group.

5.1. Evaluation with the Neighbor Distance Threshold *d*

The experiments are implemented in the synthetic datasets. The runtime of *iCPI-tree*, *CPI-tree*, *Join-based* and *Join-less* on the synthetic datasets, when the prevalence threshold *min_prev* is set as 0.3, as the neighbor distance threshold *d* increases from 18 to 28 is shown in Fig. (**5**). The iCPI-tree algorithm and the CPI-tree algorithm show less increase in the execution time with the increase of distance threshold *d*. The join-based algorithm and the join-less algo-

rithm show a rapid increase since the neighbor distance increase makes the neighbor areas larger and increase the number of co-location instances.

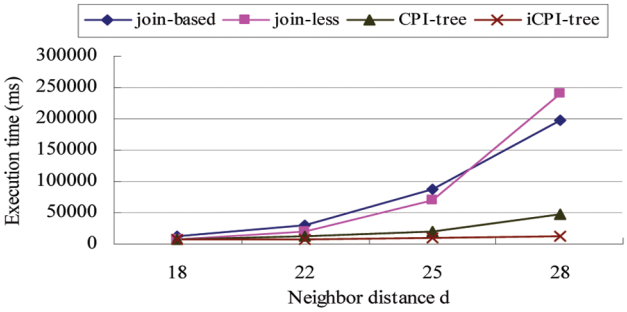

Fig. (5). Evaluation with the *distance d* over the synthetic dataset.

5.2. Evaluation with the Prevalence Threshold *Min_prev*

Fig. (**6**) presents the execution time of the four algorithms as a function of the prevalence threshold *Min_prev* over the synthetic dataset, while the comparison between the costs of the iCPI-tree algorithm and the CPI-tree algorithm is shown in Fig. (**7**). The neighbor distance threshold *d* is set as 25 in the experiments. The iCPI-tree shows much better performance at the lower threshold values. The reason is that the decrease of the *Min_prev* causes the number and the size of prevalent co-locations to be increased, which in turn may lead to an increase in the number of co-location instances. As shown in Fig. (**7**), the CPI-tree algorithm is not affected by the threshold since the CPI-tree didn't use the threshold *Min_prev* to prune.

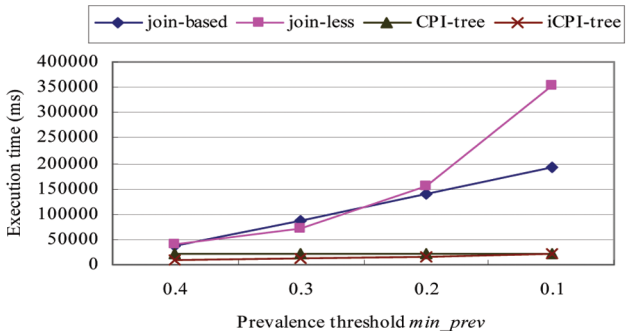

Fig. (6). Evaluation with the *Min_prev* on the synthetic dataset.

5.3. A Comparison of Generating Co-Locations Over the Size

Fig. (**8**) shows the execution times for generating the three size co-locations with the prevalence threshold *Min_prev* set to 0.3 and the neighbor distance threshold *d* set to 25 in the synthetic dataset. In the figure, the first column reports the execution time needed to discover co-locations of size 2. As can be seen, the iCPI-tree method is much faster than the join-based method and the join-less method for generating size 3 and size 4 co-locations. Thus, the iCPI-tree

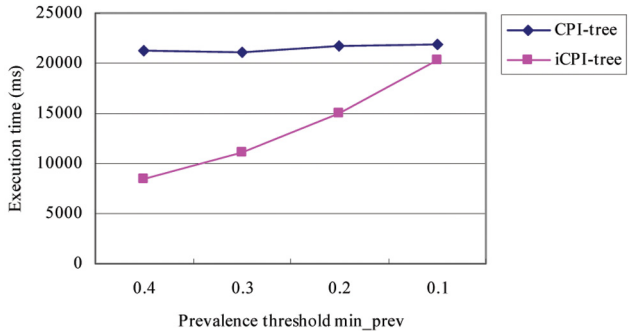

Fig. (7). Comparison of the CPI-tree and the iCPI-tree with the *Min_prev* on the synthetic dataset.

algorithm is expected to achieve the best performance when the size of co-locations becomes larger.

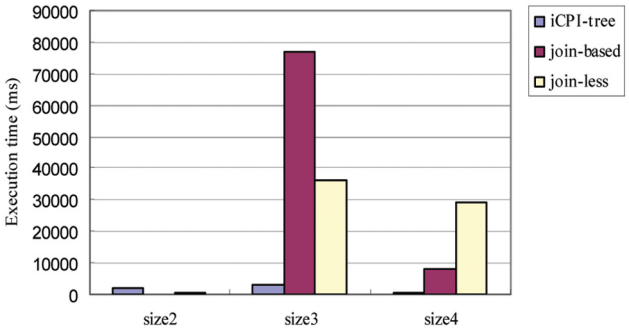

Fig. (8). Comparison of generating size 2, size 3 and size 4 prevalent co-locations on the synthetic dataset.

5.4. Experiment on a Real Dataset

The mining result over a real dataset, a plant distribution dataset of the "Three Parallel Rivers of Yunnan Protected Areas" area, is shown in Fig. (**9**), while the comparison between the costs of the iCPI-tree algorithm and the CPI-tree algorithm is shown in Fig. (**10**). In this experiment, the neighbor distance threshold *d* set to 1500, and the prevalence threshold *Min_prev* set from 0.5 to 0.2. From the figure, one can see that iCPI-tree method is scalable even when there are many table instances. Fig. (**11**) presents the distribution of the plant data in the real dataset.

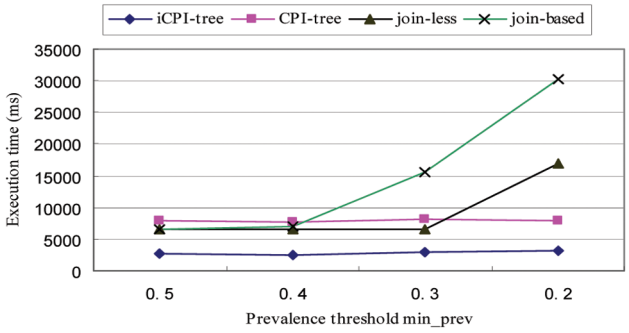

Fig. (9). A comparison using a plant distribution dataset.

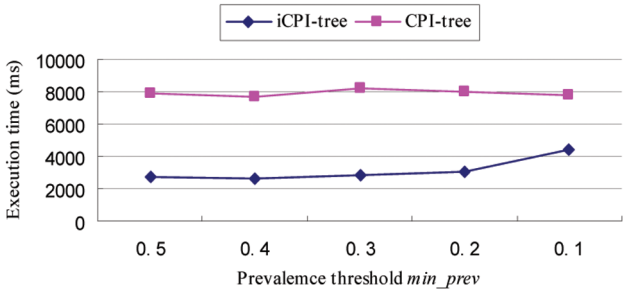

Fig. (10). A comparison of iCPI-tree and CPI-tree algorithm using a plant distribution dataset.

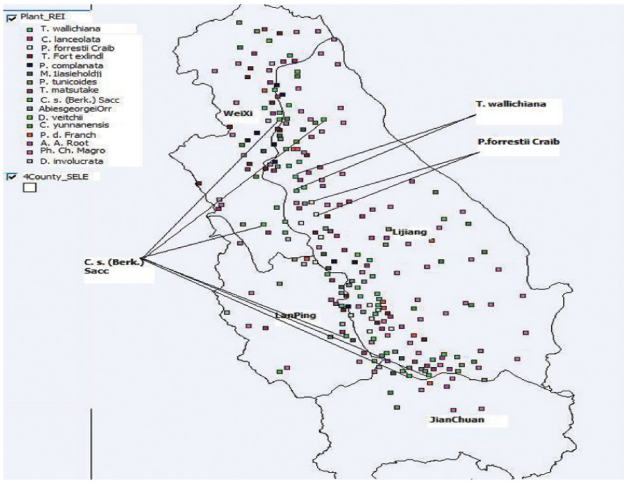

Fig. (11). An example of the distribution of plant data.

5.5. Effect of Data Density

To test the performance of the iCPI-tree algorithm against the data density, the synthetic datasets are used with the *Min-Prev* is set to *0.3*, the neighbor distance threshold *d* is 20, and the number of instances ranges from 3K to 20K. The result is shown in Fig. (**12**), which shows that the iCPI-tree algorithm has better performance to large dense datasets.

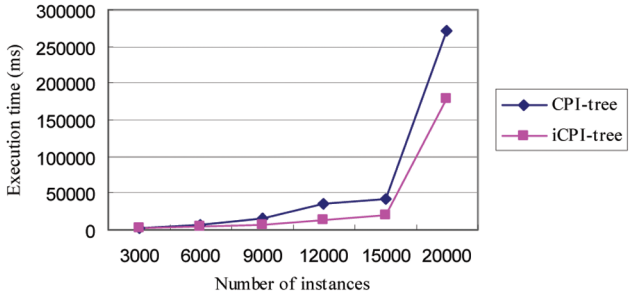

Fig. (12). Effect of data density.

6. CONCLUSION AND FUTURE WORK

In this paper, a new join-less co-location mining algorithm, which can rapidly generate spatial co-location table instances based on the i*CPI-Tree* construction materialized neighborhood relationship between spatial instances, was proposed. The algorithm is efficient since it does not require expensive spatial joins or instance join for identifying co-location table instances. The experimental results show the new method outperforms the join-based method, the join-less method and the CPI-tree method in the synthetic and the real datasets. As future work, the applications studying of co-location patterns mining is an important work. And treat with the redundant co-location rules and maximal co-location patterns mining will be significant works in the future work as well.

ACKNOWLEDGEMENTS

This research is supported by the National Natural Science Foundation of China (No. 60463004).

REFERENCES

[1] R. Agarwal and R. Srikant, "Fast algorithms for Mining association rules," In *Proceeding of Int'l Conference on Very Large Databases (VLDB)*, 1994, pp. 487-499.

[2] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, 2nd ed., Beijing, China: China Machine Press, 2006, pp. 600-607.

[3] S. Shekhar, P. Zhang, Y. Huang and R. Vatsavai, "Trends in Spatial Data Mining," in *Data Mining: Next Generation Challenges and Future Directions*, H. Kargupta, A. Joshi, K. Sivakumar and Y. Yesha, Eds., Menlo Park, CA: AAAI/MIT Press, 2004.

[4] Y. Morimoto. "Mining Frequent Neighboring Class Sets in Spatial Databases," In *Proceeding of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2001, pp. 353-358.

[5] K. Koperski and J. Han, "Discovery of Spatial Association Rules in Geographic Information Databases," In *Proc. of Int'l Symposium on Large Spatial Data bases*, 1995, pp. 47-66.

[6] L. Wang, K. Xie, T. Chen and X. Ma, "Efficient discovery of multilevel spatial association rule using partition," *Information and Software Technology (IST)*, vol. 47, no. 13, pp. 829-840, 2005.

[7] Y. Huang, S. Shekhar and H. Xiong, "Discovering colocation patterns from spatial data sets: a general approach," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, pp. 1472-1485, 2004.

[8] S. Shekhar and Y. Huang, "Co-location Rules Mining: A Summary of Results," In *Proceeding of International Symposium on Spatio and Temporal Database (SSTD)*, 2001, pp. 236-240.

[9] J. S. Yoo and S. Shekhar, "A partial Join Approach for Mining Co-location Patterns," *In Proceeding of the 12th Annual ACM International Workshop on Geographic Information Systems*, 2004, pp. 241-249.

[10] J. S. Yoo, S. Shekhar, and M. Celik, "*A Join-Less Approach for Co-Location Pattern Mining: A Summary of Results*," ICDM, Houton, Texas, 2005, pp. 813-816.

[11] L. Wang, Y. Bao, J. Lu, J. Yip, "A New Join-less Approach for Co-location Pattern Mining," In *Proceeding of the IEEE 8th International Conference on Computer and Information Technology (CIT2008)*, 2008, pp. 197-202.

[12] Y. Huang, J. Pei, and H. Xiong, "Mining co-location patterns with rare events from spatial data sets," *GeoInformatica*, vol. 10, pp. 239-260, 2006.

[13] F. Verhein, G. Al-Naymat, "Fast Mining of Complex Spatial Co-location Patterns using GLIMIT," In *Proceeding of the 2007 International Workshop on Spatial and Spatio-temporal Data Mining (SSTDM'07) in cooperation with The 2007 IEEE International Conference on Data Mining (ICDM'07)*, 2007, pp. 679-984.

[14] F. Verhein and S. Chawla, "Geometrically inspired itemset mining," In *Proceeding of the 6th IEEE International Conference on Data Mining (ICDM 2006)*, 2006, pp. 655-666.

[15] M. Celik, J. M. Kang and S. Shekhar, "Zonal Co-location Pattern Discovery with Dynamic Parameters," In *Proceeding of the 7th IEEE International Conference on Data Mining (ICDM '07)*, 2007, pp. 433-438.

[16] M. H. Alsuwaiyel, *Algorithms Design Techniques and Analysis*, Beijing, China: Publishing House of Electronics Industry, 2004, pp. 364-395.

[17] M. Berg, M. Kreveld, O. M and O. Schwarzkopf, *Computational Geometry*. US: Springer, 2000, pp. 302-350.

[18] S. Shekhar and S. Chawla, *Spatial Databases: A Tour*. NJ: Prentice Hall, 2003.

Received: February 15, 2009 Revised: May 25, 2009 Accepted: June 03, 2009