

# A Detection of Multi-level Co-location Patterns based on Column Calculation and DBSCAN Clustering

Ting Yang<sup>1</sup>, Lizhen Wang<sup>2</sup>(✉), Lihua Zhou<sup>1</sup> and Hui Chen<sup>2</sup>

<sup>1</sup> School of Information Science and Engineering, Yunnan University, Kunming 650091, China

<sup>2</sup> Dianchi College, Kunming 650228, China  
lzhwang@ynu.edu.cn

**Abstract.** Spatial co-location pattern mining aims to detect implicit, instructive, or predictive patterns. Due to the heterogeneity of spatial data distribution, we divide spatial co-location patterns into global and local patterns. Multi-level co-location pattern mining is the method that simultaneously detects them. However, existing mining methods have issues such as high time consumption and poor quality in the discovery of localities of local co-location patterns. In order to enhance the mining efficiency of co-location patterns, this study proposes to use the column calculation method instead of the traditional row calculation method in co-location pattern mining. Meanwhile, to improve the discovery quality of localities, this study proposes to use DBSCAN clustering method to detect the localities of local co-location patterns. Experimental results on synthetic and real-world datasets demonstrate that the proposed method improves the time efficiency by several times orders of magnitude compared to existing methods, along with an improvement in the quality of the identified localities.

**Keywords:** Spatial data mining, Multi-level co-location pattern, Column Calculation, DBSCAN.

## 1 Introduction

Modern society is undergoing a wave of digital transformation. Technologies like satellites, remote sensing, and the Internet of Things are developing rapidly. This leads to explosive growth in spatial data. Such vast amounts of spatial data contain infinite opportunities and challenges. Thus, spatial data mining has emerged as a response. Its task is to discover valuable patterns and knowledge from these spatial datasets. Spatial co-location pattern mining is a significant branch of spatial data mining. It aims to discover implicit, instructive, or predictive patterns [1]. A spatial co-location pattern is a subset of the spatial feature set, whose instances are prevalently located nearby in space [2]. For example, the spatial co-location pattern {gas station, convenience store, public restroom} indicates that these three often coexist in close geographic proximity. Spatial co-location pattern mining can not only deepen our understanding of geospatial but also provide substantial support for decision-making, playing an important role in public safety [3], geosciences [4], and urban planning [5].

Traditional spatial co-location patterns are mainly categorized into global co-location patterns (GCPs) prevalent globally and local co-location patterns (LCPs) prevalent in localities. To solve the problem that the localities containing LCPs are unknown a priori, scholars have proposed multi-level co-location pattern mining methods aimed at simultaneously mining both GCPs and LCPs. The methods include two steps. First, mine GCPs. Second, consider globally non-prevalent co-location patterns as candidate LCPs. Then, identify their localities, in which the patterns are prevalent. Existing methods [6-7] typically use traditional pattern mining approaches, such as join-less or join-based, to detect GCPs. Traditional methods primarily calculate the participation index of co-location patterns by constructing table instances, and then assess the prevalence of the co-location patterns. However, creating complete table instances is sufficient but not necessary for calculating the participation index. Moreover, existing methods either adopt adaptive clustering techniques [6] or clustering methods based on a percentile estimation formula that removes extremely long edges [7] to discover the localities of LCPs. The former clusters the centroids of row instances of the co-location pattern, which is time-consuming. The latter removes edges longer than a fixed percentage, which does not adapt well to data with different distributions, resulting in poor clustering outcomes. To solve the aforementioned issues, this study proposes a multi-level co-location pattern mining method based on column calculation and DBSCAN clustering, aimed at enhancing the efficiency and quality of multi-level co-location pattern mining. This study's contributions are threefold:

1. Addressing the unnecessary time expenditure in existing methods that use traditional co-location pattern mining approaches by constructing complete table instances to detect GCPs, this study proposes using column calculation to directly obtain participating instances of co-location patterns.
2. To address the issue of significant time costs and poor clustering results in existing methods for mining localities of LCPs, this study introduces the use of DBSCAN clustering to discover localities of LCPs.
3. Experiments on synthetic and real-world datasets demonstrate that the proposed method improves efficiency by several times to several orders of magnitude compared to existing methods and exhibits even better performance in identifying localities of LCPs.

## 2 Related Work

### 2.1 Global Co-location Pattern Mining

The concept of colocation patterns was first introduced by Shekhar et al. [2], and they were the first to propose a complete Apriori-like spatial colocation pattern mining framework based on the concept of participation index, which level-wise mines co-location patterns that are prevalent across the entire geographical space. Subsequently, researchers actively explored various methods, including improving the materialization method of neighbor relationships, optimizing pruning strategies [8-11] and introducing parallel techniques [12-13], to enhance the mining efficiency of co-location patterns. In

particular, Yang et al. [1] proposed a heuristic column calculation method, which does not generate table instances but directly searches for participating instances to calculate the participation index, which significantly improves the efficiency of co-location pattern mining. All of the above methods follow the bottom-up mining framework proposed by Shekhar et al. which suffers from a large number of repetitive searches, so some scholars [14-16] have proposed clique-based top-down mining approaches. These approaches first calculate the maximal spatial cliques based on the neighbor relationships between instances and then discover the prevalent spatial co-location patterns from the maximal cliques.

## 2.2 Local Co-location Pattern Mining

Celik et al. [17] proposed the use of a quad-tree index structure to divide the global area, and then mined prevalent co-location patterns in each sub-region. This method can address the challenges of dynamic parameters and achieve dynamic mining of LCPs, but determining the number and size of partitions requires complex prior knowledge. Ding et al. [18] proposed a clustering method based on multi-resolution grids to partition areas, aiming to find interesting localities based on a local interest measure, but this also faces the problem of requiring complex prior knowledge. To address the above issues, Qian et al. [19] introduced a hierarchical partitioning method based on the k-nearest neighbor graph (KNNG), treating each KNNG as an independent locality to mine co-location patterns. This method replaces the neighbor distance threshold with a distance variation coefficient to drive the mining process, which can automatically determine the k value for each locality.

## 2.3 Multi-level Co-location Pattern Mining

Deng et al. [6] proposed a multi-level co-location pattern mining method, ML, which initially uses a join-based approach to discover GCPs, and then utilizes an adaptive spatial clustering method to discover localities of LCPs. Subsequently, Liu et al. [7] built upon Deng's research to propose a multi-level co-location pattern mining method based on natural neighborhoods, named ML-Miner. This method defines natural neighborhoods based on the formation mechanisms of co-location patterns and the local distribution of spatial features. It adaptively constructs neighbor relationships for instances with different features. The method first mines GCPs based on neighbor relationships using the joinless approach, and then designs a multi-directional method for candidate LCPs to detect their localities.

# 3 Related Concepts

## 3.1 Spatial Co-location Pattern

Spatial **feature**  $f_i$  refers to the type of different spatial objects in space. A spatial **instance** of feature  $f_i$  denotes a spatial object with feature type  $f_i$ , represented by the triple  $\langle \text{instance id, feature, location} \rangle$ . The **set of spatial features** is the collection of all

spatial features, denoted as  $F=\{f_1, f_2, \dots, f_n\}$ . The set of spatial instances is the collection of all spatial objects, denoted as  $I=\{i_1, i_2, \dots, i_m\}$ . If the distance between two instances is less than or equal to a given **neighbor distance threshold**  $d$ , then these two instances are said to have a **neighbor relationship**, denoted as  $R(i_x, i_y) \Leftrightarrow \text{distance}(i_x, i_y) \leq d$ , where  $\text{distance}(i_x, i_y)$  represents the distance between instances  $i_x$  and  $i_y$ . A **spatial co-location pattern**  $C=\{f_1, f_2, \dots, f_k\} (k \geq 2)$  is a subset of the spatial feature set, whose instances prevalently appear in neighbor in space, and the number of features contained in  $C$  is the **size** of  $C$ . The instance set  $RI=\{i_1, i_2, \dots, i_k\}$  represents a **row instance** of  $C$ , satisfying two conditions: (1) the number of instances in  $RI$  is the same as the size of  $C$ , and  $RI$  includes all features of  $C$ ; (2) any two instances have a neighbor relationship. For example, the three instances connected by yellow triangles in Fig. 1 constitute a row instance of a size-3 co-location pattern. The collection of  $RI$ s is known as the **table instance** of pattern  $C$ , denoted as  $TIns(C)$ . Instances of  $f_i$  in  $TIns(C)$  are called **participating instances** of  $f_i$ , and the set of participating instances for  $f_i$  is denoted as  $PIns(C, f_i)=\{i_x | i_x \in TIns(C) \wedge i_x.\text{feature}=f_i\}$ . The **participation ratio**  $PR(C, f_i)$  of  $f_i$  in pattern  $C$  is defined as the ratio of the number of participating instances of  $f_i$  to the total number of instances of  $f_i$ , expressed as:

$$PR(C, f_i) = \frac{|PIns(C, f_i)|}{|I_i|} \quad (1)$$

Here,  $I_i$  denotes the set of instances of  $f_i$ , and  $|\cdot|$  denotes the number of elements in the set.

The **participation index**  $PI(C)$  is defined as the minimum participation ratio of features in pattern  $C$ , expressed as:

$$PI(C) = \min_i \{PR(C, f_i)\} \quad (2)$$

The concept of participation index was proposed by Shekhar et al. [2] to measure the prevalence of pattern  $C$ . Given a **prevalence threshold**  $\text{min\_prev}$ , if  $PI(C) \geq \text{min\_prev}$ ,  $C$  is termed a prevalent co-location pattern. Since pattern  $C$  is prevalent on a global scale, it is referred to as a globally prevalent co-location pattern.

Due to the heterogeneity of spatial data distribution, co-location patterns that are non-prevalent on a global scale may be prevalent in certain specific localities. Therefore, the **local co-location pattern** is defined as  $LCP = (C, \text{locality})$ , where  $\text{locality}$  is a subset of the set of spatial instances. Analogous to the participation ratio, the **local participation ratio**  $LPR((C, \text{locality}), f_i)$  is defined as the ratio of the number of participating instances of feature  $f_i$  to the total number of instances of  $f_i$  in the current locality, expressed as:

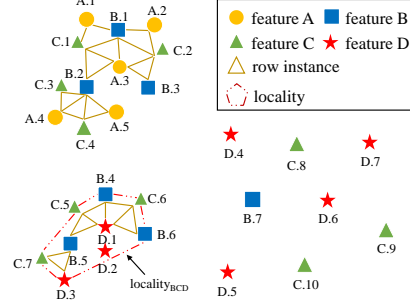


Fig. 1. An example of spatial datasets.

$$LPR((C, locality), f_i) = \frac{|PIns((C, locality), f_i)|}{|\text{\#instances of } f_i \text{ in } locality|} \quad (3)$$

where  $PIns((C, locality), f_i)$  denotes the set of participating instances of the feature  $f_i$  in the locality of the  $C$ .

The **local participation index**  $LPI((C, locality))$  is defined as the minimum value of the local participation ratio of features in the  $LCP$ , denoted as:

$$LPI((C, locality)) = \min_i^k \{LPR((C, locality), f_i)\} \quad (4)$$

The concept of local participation index was introduced by Mohan et al. [3], used to measure the prevalence of LCPs within localities. Given a  $LCP_i = (C_i, locality_m)$  and a prevalence threshold  $min\_prev$ , if  $LPI(LCP_i) \geq min\_prev$ , then  $LCP_i$  is a locally prevalent co-location pattern in  $locality_m$ . For example, in Fig. 1, the participation index  $PI(\{B, C, D\}) = \min\{3/7, 3/10, 2/7\} = 2/7$ ; if  $min\_prev$  is set to 0.3, then the pattern  $\{B, C, D\}$  is not prevalent globally, but  $LPI(\{B, C, D\}, locality_{BCD}) = \min\{1, 1, 2/3\} = 2/3 \geq min\_prev$ , indicating that  $\{B, C, D\}$  is a locally prevalent co-location pattern in the locality  $locality_{BCD}$ .

### 3.2 Column Calculation

The neighbor set of instance  $i_x$  consists of all instances that have a neighbor relationship with it, denoted as  $Neigh(i_x) = \{i_y | i_y \in I, R(i_x, i_y)\}$ . Furthermore, instances within  $Neigh(i_x)$  are grouped by feature, resulting in the grouped neighbor set of  $i_x$  under feature  $f_i$ , denoted as  $groupN(i_x, f_i) = \{i_y | i_y \in I, i_y \in Neigh(i_x), i_y.feature = f_i\}$ . For example, in Fig. 1, for instance  $B.2$ , there is  $groupN(B.2, C) = \{C.1, C.2, C.4\}$ .

According to reference [1], the participating instances of a co-location pattern  $C$  are certainly included in the participating instance set of its sub-patterns, thus the candidate participating instance set of  $f_i$  in the size- $k$  co-location pattern  $C$  is the intersection of the participating instance sets of all size- $(k-1)$  sub-patterns of  $C$  that include  $f_i$ , represented as:

$$CPIns(C, f_i) = \bigcap_{SC \in C_{k-1}^{f_i}} PIns(SC, f_i) \quad (5)$$

Here,  $C_{k-1}^{f_i}$  denotes the set of all size- $(k-1)$  sub-patterns of  $C$  that include  $f_i$ .

For the instance  $i_x$  of feature  $f_i$  in the co-location pattern  $C$ , the set of instances of other features in  $C$  that may form row instances with  $i_x$  is called the instance search space on feature  $f_i$ , denoted as:

$$ISS(i_x, f_i, C) = \begin{cases} groupN(i_x, f_i) \cap PIns(C, f_i) & PIns(C, f_i) \text{ known} \\ groupN(i_x, f_i) \cap CPIns(C, f_i) & PIns(C, f_i) \text{ unknown} \end{cases} \quad (6)$$

where by searching for instances within the candidate participating instance set and verifying whether they form row instances with instances in their search space, the participating instance set for  $f_i$  ( $f_i \in C$ ) can be obtained.

## 4 Multi-level Co-location Pattern Mining

Based on the fundamental concepts and principles of the column calculation and the DBSCAN clustering method, this section proposes the ML-CCDB method (Algorithm 1) for mining multi-level co-location patterns. In this algorithm, step 1 involves acquiring the neighbor relationships between instances and storing the grouped neighbors of each instance under different features. Step 2 initializes each spatial feature as a size-1 prevalent co-location pattern and sets  $k=2$ , beginning the iteration. Steps 3-16 start from size-2 and level-wise mine GCPs and LCPs. Algorithm 1 ultimately returns all GCPs and LCPs, and the localities of LCPs.

---

### Algorithm 1 ML-CCDB

---

Input:  $I, F, d, min\_prev$ .

Output:  $P$

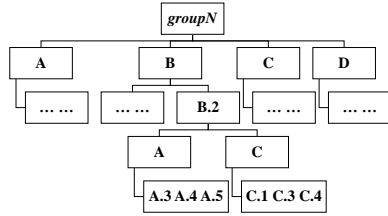
1.  $groupN = get\_groupN(d, I, F)$
  2. Initialize  $P_1 = F, k = 2$
  3. while  $P_{k-1} \neq \emptyset$  do
  4.    $P_k = \emptyset, C_k = gen\_candidate\_pattern(P_{k-1})$
  5.   for each  $C$  in  $C_k$  do
  6.      $PIns(C) = get\_PIns(C, groupN)$
  7.      $PI = calculate\_PI(C, PIns(C))$
  8.     if  $PI \geq min\_prev$  do
  9.        $C.level = Global$
  10.       $P_k = P_k \cup \{C\}$
  11.     else do
  12.        $C.localities = get\_localities(d, k, C, PIns(C), min\_prev)$
  13.       if  $C.localities \neq \emptyset$  then
  14.           $C.level = Local$
  15.           $P_k = P_k \cup \{C\}$
  16.       $k = k + 1$
  17. return  $(P_2, ..., P_{k-1})$
- 

In Algorithm 1, Step 1 constructs the  $groupN$  based on the neighbor distance threshold  $d$  using the plane sweep method [20], and stores the set in a 4-layer hash structure as shown in Fig. 2. Step 4 generates size- $(k+1)$  candidate co-location patterns from size- $k$  prevalent co-location patterns. Step 6 employs the column calculation method to obtain the participating instance set for candidate co-location patterns, with its pseudocode presented in Algorithm 2. Step 7 calculates the participation index of the candidate pattern, and if the participation index is greater than or equal to the prevalence threshold  $min\_prev$ , the pattern is identified as a globally prevalent co-location pattern. Otherwise, Step 12 uses the  $get\_localities$  method to mine its localities, with pseudocode as shown in Algorithm 3.

**Algorithm 2** get\_PInsInput:  $C, groupN$ Output:  $PIns(C)$ 

1. for each  $f$  in  $C$  do
2.    $CPIns(C, f) = \text{get\_CPIns}(PIns(C))$
3.    $PIns(C, f) = \emptyset$
4.   for each  $i$  in  $CPIns(C, f)$  do
5.      $ISS(i, f, C) = \text{gen\_ISS}(groupN(i, f), CPIns(C, f))$
6.     if  $i$  not in  $PIns(C, f)$  then
7.        $RI = \text{search\_RI}(i, C)$
8.       if  $RI \neq \emptyset$  then
9.          for each  $i'$  in  $RI$  do
10.            $PIns(C, i'.feature) = PIns(C, i'.feature) \cup \{i'\}$
11. return  $PIns(C)$

In Algorithm 2, Step 2 employs Eq. 5 to obtain the candidate participating instances of the co-location pattern  $C$ , which are the intersections of the participating instances of its sub-patterns. Step 5 utilizes Eq. 6 to acquire the search space for the candidate participating instances. According to Eq. 6, by using the hash structure depicted in Fig. 2 to store the grouped neighbor sets, Step 5 can retrieve  $ISS(i, f, C)$  with a time complexity of  $O(1)$ , thereby reducing the time and enhancing the efficiency of obtaining the participating instances of candidate co-location patterns through the column calculation method. Step 7 adopts a backtracking approach to quickly search for a row instance that includes the candidate participating instance  $i$ ; if this row instance is not empty, then  $i$  and other instances within the row instance are marked as participating instances. The detailed steps of the backtracking process can be found in reference [1].



**Fig. 2.** Hash structure for  $groupN$ . The figure shows part of the  $groupN$  for instances B.2 in Fig. 1.

**Algorithm 3** get\_localitiesInput:  $d, k, C, PIns(C), min\_prev$ ;Output: localities of  $C$ 

1.  $C.localities = \emptyset$
2.  $clusters = \text{DBSCAN}(d, k, PIns(C))$
3. for each  $cluster$  in  $clusters$  do
4.    $boundary = \text{get\_convex\_hull}(cluster)$
5.    $locality = \text{get\_locality}(boundary)$
6.    $LPI = \text{calculate\_LPI}(C, locality)$
7.   if  $LPI \geq min\_prev$  then
8.      $C.localities = C.localities \cup \{locality\}$
9. return  $C$

In Algorithm 3, Step 2 employs the DBSCAN clustering algorithm to cluster the participating instances. By setting the clustering parameter radius to the neighbor distance threshold  $d$ , and the minimum number of points to the size  $k$  of the co-located patterns, all participating instances can be identified as core points, ensuring that instances with neighbor relationships are directly density-reachable to each other. This guarantees that instances forming a row instance belong to the same cluster, thus ensuring that no neighbor relationships of the co-located patterns are missed. For each cluster, Steps 4-9 first determine the convex hull of the cluster, and then assess the prevalence of pattern  $C$  within the locality enclosed by the convex hull, and if the local participation index of

pattern  $C$  in the locality is not less than the prevalence threshold, then the locality is identified as a locality for pattern  $C$ .

## 5 Experimental evaluation

### 5.1 Experimental design

**Real datasets.** In the experiments, a real-world datasets were utilized, namely the Shenzhen POI dataset.

**Synthetic datasets.** The synthetic datasets were generated using a generator similar to the one described in reference [7], with the main steps as follows:

1. Divide a  $L \times L$  study area into four  $L/2 \times L/2$  sub-areas. Select three of these sub-areas to generate data points as data domains and one sub-area to generate noise points as a noise domain. In each data domain, generate a parent point at its center.
2. For each parent point, generate  $n_{child}$  child points uniformly distributed around the parent point within a buffer zone with a radius of  $r_{parent}$ , then remove all parent points.
3. For each child point, generate  $n_{grandchild}$  grandchild points with  $n$  different feature types, uniformly distributed around the child point within a buffer zone with a radius of  $r_{child}$ , then remove all child points.
4. In the noise domain, randomly generate  $n_{noise}$  points belonging to a certain feature.

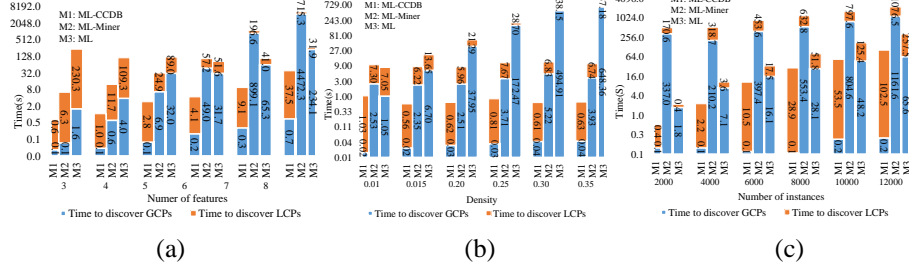
In the synthetic data generator,  $r_{parent}$  and  $n_{child}$  control the density of spatial instance distribution,  $r_{child}$  controls the distance between instances of different features,  $n_{grandchild}$  controls the number of features, and  $n_{noise}$  controls the prevalence of the pattern.

**Comparative methods.** The comparative methods include the ML [6] method and the ML-Miner [7] method mentioned in Section 2.3.

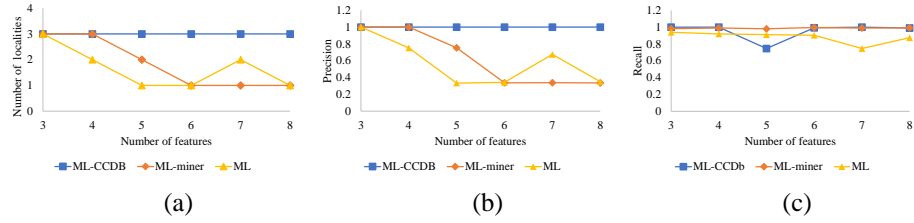
**Evaluation metrics.** The mining quality of LCPs was evaluated using the assessment method described in reference [6], with precision and recall. Defines as  $precision = TP / (TP + FP)$ ,  $recall = TP / (TP + FN)$ , Where  $TP$  (True Positives) refers to the correctly identified number in the real prevalent localities,  $FP$  (False Positives) refers to the incorrectly identified number in the real prevalent localities, and  $FN$  (False Negatives) refers to the number not correctly identified in the predetermined localities.

### 5.2 Influence of the number of spatial features

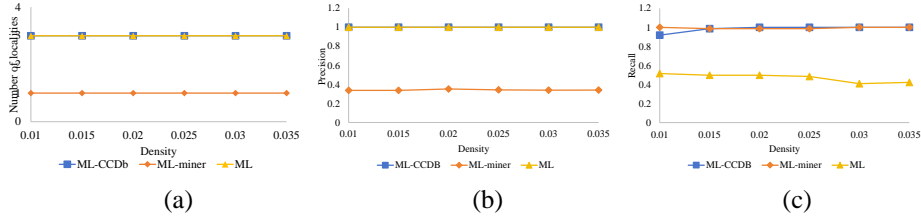
This experiment tests the influence of the number of spatial features on multi-level co-location pattern mining methods using synthetic datasets. To maintain the same density  $\rho$  across different datasets, we set  $n_{grandchild}$  within [3, 8],  $L=1200$ ,  $n_{child}=850/n_{grandchild}/3$ ,  $r_{parent}=(n_{child} \times n_{grandchild})/(\rho \times 3.14)$ , and  $r_{child}=25$ . For the threshold values of each method, the neighbor distance threshold  $d$  for ML-CCDB and ML methods was set to  $2 \times r_{child}$ , and the prevalence threshold  $min\_prev$  for the three methods was set to 0.3.



**Fig. 3.** The effect of different parameters on the runtime of multilevel co-location pattern mining. (a) The number of features; (b) Density of distribution of instances; (c) The number of instances.



**Fig. 4.** The evaluation of results at different feature numbers. (a) Number of localities; (b) Precision; (c) Recall



**Fig. 5.** The evaluation of results at different instance distribution densities. (a) Number of localities; (b) Precision; (c) Recall

**Table 1.** Number of participating instances or row instances

	Density					
Method	0.01	0.015	0.02	0.025	0.03	0.035
ML-CCDB(PIns)	3167	3287	3315	3315	3315	3315
ML(RI)	15178	44170	105326	210222	334948	379089

Moreover, to include both GCPs and LCPs in the synthetic data,  $n_{noise}$  was set to  $\text{int}(n_{child} \times 3 / \min\_prev - n_{child} \times 3) + 1$ , ensuring that the participation ratio of noise features in co-location patterns does not exceed  $\min\_prev$ . The experimental results are shown in Fig. 3(a) and Fig. 4.

Figure 3(a) displays the running time of the three methods on synthetic data with varying numbers of features. By observing the time taken by each method to mine GCPs, it can be found that the column computation method used by ML-CCDB

consumes almost less than one second, with minimal influence from the number of features, representing an efficiency improvement of several orders of magnitude compared to the traditional methods used by the other two methods. Additionally, the time taken by the ML-CCDB method to mine LCPs is significantly less than that of other methods. This demonstrates that the ML-CCDB method is more efficient compared to ML-Miner and ML.

Figure 4 presents the number of localities, precision, and recall of the highest size ( $k=n_{grandchild}$ ) LCPs mined by each method. It can be seen that the ML-CCDB method possesses high precision and recall, enabling it to mine co-location patterns completely and correctly. The ML-Miner and ML methods have lower precision because they cannot effectively separate the localities of LCPs in the three data domains. This is because the ML-Miner method uses a fixed method of deleting edges longer than 90% to obtain localities, and such a fixed percentage setting cannot perfectly adapt to datasets with different distributions, thus neglecting some long edges that should be deleted. The ML method also faces the issue of missing long edge identification when it opts to delete global and local long edges to detect localities. The DBSCAN clustering method used in this study can effectively distinguish closely situated clusters without the need for additional parameters from the user, thus providing better results in mining localities.

### 5.3 Influence of the density of distribution of spatial instances.

This experiment tests the impact of spatial instance distribution density on multi-level co-location pattern mining methods using synthetic datasets. To maintain the same features and number of instances across different datasets, the density  $\rho$  is set within  $[0.01, 0.035]$ ,  $n_{grandchild}=5$ ,  $L=300$ ,  $n_{child}=15$ ,  $r_{parent}=(n_{child} \times n_{grandchild})/(\rho \times 3.14)$ ,  $r_{child}=15$ , and  $n_{noise}=\text{int}(n_{child}^3/\text{min\_prev} - n_{child} \times 3)+1$ . The experimental results are shown in Fig. 3 (b), Fig. 5 and Table 1.

Figure 3(b) displays the running times of three methods on synthetic data with different instance distribution densities, while Table 1 presents the total number of participating instances and row instances during the execution of the ML-CCDB and ML methods. It can be observed that as the instance distribution density increases, there is no significant change in the running time of the ML-CCDB and ML-Miner methods. The reason is that both ML-CCDB and ML-Miner methods employ a method of clustering participating instances of patterns to detect localities of LCPs, whereas the ML method uses clustering of row instance centroids. However, with the increase in instance distribution density, the number of participating instances does not increase significantly, while the number of row instances almost doubles. Therefore, this indicates that the method of clustering participating instances to mine localities used by our method and the ML-Miner method is more efficient and universally applicable. Fig. 5 shows the number of localities, precision, and recall of the highest size ( $k=n_{grandchild}$ ) LCPs mined by each method. The ML-CCDB method still maintains high precision and recall. The performance of the ML-Miner method is similar to that of experiment 5.2, and although the ML method has high precision, it still lacks perfection in the judgment of long edges in local patterns, leading to an incomplete identification of localities of LCPs.

#### 5.4 Influence of the number of spatial instances

This experiment utilizes the Shenzhen POI dataset to test the impact of spatial instance number on multi-level co-location pattern mining methods. Specifically, datasets with varying instance number belonging to 6 different features were extracted for the experiment, setting the instance number  $n$  within the range [2000,12000] and the neighbor distance threshold  $d$  at 100. Concurrently, to enable the methods to mine both global and LCPs simultaneously, and to ensure that the number of mined LCPs is roughly equivalent to the highest size, patterns with a participation index less than 1 are considered candidate LCPs, while those with a participation index greater than 0 are deemed LCPs. The experimental results are shown in Fig. 3(c).

Figure 3(c) displays the runtime of three methods on datasets with different instance number. It can be observed that ML-CCDB still performs well in mining efficiency for both GCPs and LCPs, and the time taken to mine GCPs is almost unaffected by the increase in instance quantity. In contrast, ML-Miner and the ML method require more time when dealing with larger instance datasets. The reason is that as the number of instances increases, the neighbor relationships increase, and both joinless and join-based methods require more time to generate table instances. Simultaneously, the time taken by the three methods to mine LCPs and their localities increases with the instance number, because the clustering operations require more time as the number of instances grows.

## 6 Conclusion

Multi-level co-location pattern mining aims to simultaneously mine GCPs and LCPs. This paper analyzes the issues with existing multi-level co-location pattern mining methods and proposes the ML-CCDB method. The method uses column calculation to mine GCPs and obtain instances that participate in LCPs, then employs the DBSCAN clustering method to detect localities of LCPs. Experimental results indicate that the ML-CCDB method outperforms other multi-level co-location pattern mining methods in efficiency, enhancing it by several times, and the mined localities are also more reasonable.

**Acknowledgments.** This work was supported by the National Natural Science Foundation of China (62276227, 62266050, 62306266), the Project of Innovative Research Team of Yunnan Province (2018HC019), the Yunnan Fundamental Research Projects (202201AS070015, 202401AT070450), and the Postgraduate Research and Innovation Foundation of Yunnan University (KC-23235527).

## References

1. Yang, P., Wang, L., Wang, X., Zhou, L.: A spatial co-location pattern mining approach based on column calculation. *Sci. Sin. -Inf.* 52, 1053 (2022).

2. Huang, Y., Shekhar, S., Xiong, H.: Discovering colocation patterns from spatial data sets: a general approach. *IEEE Trans. Knowl. Data Eng.* 16, 1472–1485 (2004).
3. Mohan, P., Shekhar, S., Shine, J.A., Rogers, J.P., Jiang, Z., Wayant, N.: A neighborhood graph based approach to regional co-location pattern discovery: a summary of results. In: *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. pp. 122–132. ACM, Chicago Illinois (2011).
4. Li, J., Adilmagambetov, A., Mohomed Jabbar, M.S., Zaïane, O.R., Osornio-Vargas, A., Wine, O.: On discovering co-location patterns in datasets: a case study of pollutants and child cancers. *Geoinformatica*. 20, 651–692 (2016).
5. Yao, X., Jiang, X., Wang, D., Yang, L., Peng, L., Chi, T.: Efficiently mining maximal co-locations in a spatial continuous field under directed road networks. *Information Sciences*. 542, 357–379 (2021).
6. Deng, M., Cai, J., Liu, Q., He, Z., Tang, J.: Multi-level method for discovery of regional co-location patterns. *International Journal of Geographical Information Science*. 31, 1846–1870 (2017).
7. Liu, Q., Liu, W., Deng, M., Cai, J., Liu, Y.: An adaptive detection of multilevel co-location patterns based on natural neighborhoods. *International Journal of Geographical Information Science*. 35, 556–581 (2021).
8. 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*. pp. 241–249. ACM, Washington DC USA (2004).
9. Yoo, J.S., Shekhar, S.: A Joinless Approach for Mining Spatial Colocation Patterns. *IEEE Trans. Knowl. Data Eng.* 18, 1323–1337 (2006).
10. Hu, Z., Wang, L., Tran, V., Chen, H.: Efficiently mining spatial co-location patterns utilizing fuzzy grid cliques. *Information Sciences*. 592, 361–388 (2022).
11. Wu, P., Wang, L., Zou, M.: A maximal ordered ego-clique based approach for prevalent co-location pattern mining. *Information Sciences*. 608, 630–654 (2022).
12. Yoo, J.S., Boulware, D., Kimmey, D.: Parallel co-location mining with MapReduce and NoSQL systems. *Knowl Inf Syst.* 62, 1433–1463 (2020).
13. Andrzejewski, W., Boinski, P.: Efficient spatial co-location pattern mining on multiple GPUs. *Expert Systems with Applications*. 93, 465–483 (2018).
14. Bao, X., Wang, L.: A clique-based approach for co-location pattern mining. *Information Sciences*. 490, 244–264 (2019).
15. Zhang, S., Wang, L., Tran, V.: CPM-MCHM: A spatial co-location pattern mining approach based on maximal clique and hash table. *Chinese Journal of Computers*, 45(3): 526–541(2022).
16. Li, J., Wang, L., Tran, V., Li, J., Jiang, X.: Fast Mining Prevalent Co-location Patterns Over Dense Spatial Datasets. In: *Spatial Data and Intelligence*. pp. 179–191. Springer Nature Switzerland, Cham (2023).
17. Celik, M., Kang, J.M., Shekhar, S.: Zonal Co-location Pattern Discovery with Dynamic Parameters. In: *Seventh IEEE International Conference on Data Mining (ICDM 2007)*. pp. 433–438. IEEE, Omaha, NE, USA (2007).
18. Ding, W., Eick, C.F., Yuan, X., Wang, J., Nicot, J.-P.: A framework for regional association rule mining and scoping in spatial datasets. *Geoinformatica*. 15, 1–28 (2011).
19. Qian, F., Chiew, K., He, Q., Huang, H.: Mining regional co-location patterns with kNNG. *J Intell Inf Syst.* 42, 485–505 (2014).
20. Ouyang, Z., Wang, L., Wu, P.: Spatial Co-Location Pattern Discovery from Fuzzy Objects. *Int. J. Artif. Intell. Tools*. 26, 1750003 (2017).