Received: 24 November 2023 | Revised: 25 January 2024 | Accepted: 7 February 2024 | Published online: 18 February 2024

RESEARCH ARTICLE

Journal of Data Science and Intelligent Systems
2025, Vol. 3(3) 206–215
DOI: 10.47852/bonviewJDSIS42022134

BON VIEW PUBLISHING

Fuzzy Regional Co-location Pattern Mining Based on Efficient Density Peak Clustering and Maximal Fuzzy Grid Cliques

Tao Zhou[1], Lizhen Wang[2,*], Dongsheng Wang[1] and Vanha Tran[3]

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

[2]*Dianchi College, China*

[3]*FPT University, Vietnam*

Abstract: Due to the heterogeneity of data distribution in real life and the spatial autocorrelation among spatial instances, traditional spatial co-location pattern mining methods tend to ignore valuable information specific to local regions. To address the limitation, regional co-location pattern mining has been proposed to find patterns that may be hidden within local regions. In this paper, a fuzzy regional co-location pattern mining framework based on efficient density peak clustering and maximal fuzzy grid cliques is presented. By incorporating a grid-splitting method and fuzzy theory, an efficient density peak clustering algorithm is proposed to divide the global area into distinct local regions. Furthermore, we propose a method to materialize the neighbor relationships between instances based on the maximal fuzzy grid cliques and parallelize the clustering process to improve the algorithm efficiency. Experimental results show that the proposed algorithm can not only reduce the time consumption by about 40% but also mine meaningful patterns with tighter instance distributions.

Keywords: spatial data mining, fuzzy regional co-location pattern, parallelizing density peak clustering, maximal fuzzy grid cliques

1. Introduction

As a crucial branch of data mining, the spatial data mining aims to extract valuable insights and knowledge from the massive spatial data. One significant aspect of spatial data mining is the spatial co-location pattern (SCP) mining. The set of all features in a given space is denoted as $F = \{f_1, f_2, \ldots, f_m\}$, and the set of all instances in the space is denoted as $I = \{e_1, e_2, \ldots, e_n\}$. The SCP mining focuses on revealing the distribution rules of spatial features. For instance, consider the cross-interaction among various factors influencing the risk of import and export enterprises in space to derive distribution patterns among these factors [1]. In particular, SCP mining can be used to identify spatio-temporal crime patterns with heterogeneity [2] and can also model frequently occurring events and behaviors to derive valuable information [3], while optimal cluster-based feature extraction can be used to refine the intrusion detection systems [4].

Firstly, to measure the prevalence of SCPs globally, Shekhar and Huang [5] proposed a distance-based participation index (PI). Based on this, to mine all prevalent SCPs, Huang et al. [6] proposed a Join-based algorithm and Yoo et al. [7] proposed a Partial-join and a Joinless [8] algorithm. However, these algorithms have poor scalability when processing massive data. To cope with this challenge, Chan et al. [9] proposed the Fraction-Score method to collect all possible instances based on the sharing relationship, which improved the efficiency of SCP mining. Hu et al. [10] added fuzzy theory and spatial grid on this basis, which significantly facilitates the quality and efficiency of SCP mining. Secondly, with the concept of clustering, Huang and Zhang [11] pioneered the introduction of a clustering-based framework for SCP mining, while Rodriguez and Laio [12] employed a Gaussian kernel to enhance the applicability of clustering for small datasets. Liu et al. [13] introduced local density variation metrics to effectively identify clustering centers.

The SCP mining algorithms mentioned above typically adopt a global perspective, employing absolute Euclidean distance and a single threshold to measure the neighbor relationships between instances, which assume the homogeneity of the spatial feature and instance distribution. Nevertheless, the data distribution is heterogeneous in reality, and the neighbor relationships between instances are a relative and ambiguous concept. Additionally, the autocorrelation of the space always leads to more complex neighbor relationships between instances, making the PI metric they rely on can lead to ignore valuable patterns containing rare features in local regions. The development of the regional co-location pattern (RCP) mining is a response to this issue. Suppose the global area is divided into several local regions s_i $(i = 1, 2, \ldots)$, and the RCP mining is performed in these regions, then the RCP can be denoted as $RCP = (pattern_r, s_i)$, where $pattern_r$ is a subset of the set of all spatial features in the study area and denoted as $pattern_r \subseteq F$, s_i is the local regions where the pattern

*Corresponding author: Lizhen Wang, Dianchi College, China. Email: lzhwang@ynu.edu.cn

is prevalent. Naturally, the patterns discovered may not be globally significant but are meaningful within local regions. For example, RCPs are instrumental in studying the pathways and sources of disease (such as COVID-19) transmission within a specific local region [14]. To identify RCPs in local regions, Deng et al. [15] used globally non-prevalent SCPs as candidate patterns in local regions to discover RCPs. Moreover, Jiang et al. [16] adopted the fuzzy DPC algorithm to generate local regions with ambiguous boundaries to facilitate RCP mining. However, there is uncertainty and incompleteness of data [17] during fuzzy clustering. Liu [18] proposed the CFNDC method to reduce the risk associated with uncertainty, while Liu and Letchmunan [19] proposed the DST method to optimize the clustering process for incomplete instances.

As mentioned above, although RCP mining can discover interesting patterns that cannot be found globally, traditional RCP mining still has the following challenges: first, the computational time of local density is more costly for regional division using clustering. Second, when performing RCP mining in local regions, the time consuming of identifying the participating instances of patterns is also relatively large. Third, the PI metric for measuring pattern prevalence is only focused on the number of instances involving the pattern and ignores the fuzzy and sharing relationships among instances. Based on the aforementioned background, this paper proposes a fuzzy RCP (FRCP) mining framework, which incorporates the efficient density peak clustering (EDPC) and the maximal fuzzy grid cliques (MFGCs). The main contributions of this paper include the following:

1) A regional division method based on EDPC is presented. The grid-based EDPC algorithm notably reduces the time consumption for identifying cluster centers and improves the quality of division while effectively performing regional division.
2) We propose a method to materialize the neighbor relationships between instances based on the MFGCs and parallelize the clustering process, which greatly improves the efficiency of the FRCP mining algorithm.
3) Compared to the Algorithm in Hu et al. [10] and the Algorithm in Jiang et al. [16], the algorithm presented in this paper considers the fuzzy sharing relationships between instances, which not only has a superior performing efficiency but also mine patterns with tighter distribution of instances in local regions.

The remainder of this paper is outlined as follows. Section 2 provides the EDPC-based regional division algorithm and the MFGC-based pattern mining algorithm. Section 3 analyzes the algorithm complexity. In Section 4, on synthetic and real-world datasets, the mining results of the proposed algorithm are extensively analyzed, and the significance of the divided regions is discussed.

2. Method

Traditional RCP researches neglect the consideration of fuzzy neighbor relationships among instances during the regional division and pattern prevalence calculation. In practical scenarios, instances in space can belong to multiple regions simultaneously, indicating some level of intersection among these regions. When exploring neighbor relationships, an instance may be influenced to varying degrees by instances with different features. To address these limitations, we propose a novel RCP mining framework considering fuzzy neighbor relationships between instances referred to as FRCP mining. This mining task consists of two stages: regional division and pattern mining.

2.1. Regional division algorithm based on efficient density peak clustering (EDPC)

Clustering algorithms are widely used for the regional division. In contrast to conventional clustering methods, the DPC algorithm excels at identifying cluster centers with elevated local density, allowing for the adaptive generation of diverse clusters. Inspired by the method of Jiang et al. [16], we propose an efficient DPC (EDPC) algorithm combined with grid-splitting in regional division to generate regions with ambiguous boundaries and improve the clustering efficiency. We first give the relevant concepts.

Given a collection I of instances in space, the generation of maximal fuzzy clusters involves several steps. First, for each instance $e_i \in I$, it calculates the k-nearest neighbor distances for e_i and proceeds to calculate the local density using these distances. Relative distances for e_i are then calculated, and cluster centers are selected based on these distances. Subsequently, the fuzzy membership degree of non-cluster centers to cluster centers is calculated to generate fuzzy clusters. If a fuzzy cluster is not a subset of any other fuzzy cluster, it will be included in the set of maximal fuzzy clusters, and ultimately, all maximal fuzzy clusters generated are the divided regions.

Local density ρ: given an instance $e_i \in I$, the local density of e_i is defined as:

$$\rho(e_i) = \sum_{d_k \in KNN(e_i)} \begin{Bmatrix} 1 & , d_k \le d_1 \\ 1 - \frac{d_k - d_1}{d_2 - d_1} & , d_1 < d_k < d_2 \\ 0 & , d_k \ge d_2 \end{Bmatrix} \tag{1}$$

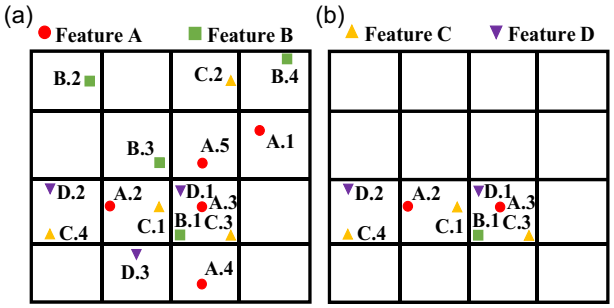
where d_k is stored in $KNN(e_i)$, denoting the Euclidean distance between e_i and the k-th nearest instance. KNN denotes the set of distances between e_i and the k-nearest neighbor instances, which is expressed as $\boldsymbol{KNN(e_i)} = \{\boldsymbol{d_1}, \boldsymbol{d_2}, \ldots, \boldsymbol{d_k}\}$. The parameters d_1 and d_2 are calculated as described in the Equations (2) and (3). Here, μ^k and S^k represent the mean and standard deviation of the k-th nearest neighbor distance of all instances, respectively.

$$d_1 = \max\left(0, \mu^k - S^k\right) \tag{2}$$

$$d_2 = \mu^k + S^k \tag{3}$$

In Figure 1(a), we set $k = 10$, then the 10-nearest instances of instance B.1 are {A.3, C.3, D.1, A.4, A.5, B.3, A.2, C.1, D.3, C.4}. By calculating the Euclidean distance between instances, we can get KNN(B.1) = {1.4, 1.4, 2, 2, 2.2, 2.2, 3.2, 4.1, 4.1, 5}. According

Figure 1
An example of spatial datasets: (a) under the grid-splitting and (b) after the grid-filtering



to this, we can also obtain the set of KNN distances for other instances, and then, the local density of each instance can be calculated.

Relative distance δ: given instances $e_i \in I$ and $e_j \in I$, the relative distance of e_i is defined as:

$$\delta(e_i) = min_{e_j:\rho_{e_i}<\rho_{e_j}} d_{ij} \tag{4}$$

where d_{ij} is the Euclidean distance between e_i and e_j, the relative distance indicates the minimum distance from e_i to instance e_j with a higher density than it. The larger the relative distance is, the more likely e_i is to be an independent cluster center.

Maximal fuzzy cluster: given an instance $e_i \in I$, The fuzzy membership degree of any non-cluster center instance e_i to a cluster center instance c_j is denoted as $u_{ij}(e_i, c_j)$ and is defined as:

$$u_{ij}(e_i, c_j) = \sum_{q=1}^{CC} \left(\frac{d_{ij}}{d_{iq}} \right)^{-\frac{2}{r-1}} \tag{5}$$

where CC is the number of cluster center instances, the distance between a non-cluster center instance e_i and a cluster center instance c_j is represented by d_{ij}, d_{iq} denotes the distance between e_i and the q-th cluster center, r denotes the smoothing parameter consistently set to $r = 2$ in the common research.

After the instances are selected with higher local density and relative distance as cluster centers, the fuzzy membership degree of non-cluster center instances to cluster centers is calculated, and if it is not less than a given threshold η, the instance is assigned to the fuzzy cluster to which it belongs, where the threshold of fuzzy membership degree is set to the reciprocal of the number of cluster centers, i.e., $\eta = 1/|CC|$. If the fuzzy cluster is not a subset of any other fuzzy cluster, then it is a maximal fuzzy cluster, which represents a sub-region in space. Eventually, the regional division of the space can be completed by utilizing the maximal fuzzy clusters.

As shown in Figure 1(a), according to the calculation method of local density and relative distance, we obtain the sole cluster center B.1 and $v_1 = 1$. According to the Equation (5), the fuzzy membership degree of all non-cluster center instances to B.1 is calculated to be 1. Also, we obtain the threshold of fuzzy membership degree is 1. In other words, instances with a fuzzy membership degree not less than the threshold are assigned to the cluster centered at B.1. Furthermore, because the cluster generated by B.1 is not a subset of any other fuzzy cluster, it is a maximal fuzzy cluster, i.e., the maximal fuzzy cluster centered at B.1 is B.1 = {A.1, A.2, A.3, A.4, A.5, B.2, B.3, B.4, C.1, C.2, C.3, C.4, D.1, D.2, D.3}.

To expedite the identification of clustering centers, we draw inspiration from the approach presented in Xu et al. [20] and introduce grid-splitting to enhance the division process. This involves dividing the whole area into grids and quantifying grid density by counting the number of instances within each grid. Since grids with sparse local densities are impossible to include clustering centers, filtering them out allows us to acquire an **efficient density peak clustering (EDPC)** algorithm. Next, we will elucidate how the EDPC algorithm improves the efficiency of cluster center identification and maximal fuzzy cluster generation.

First, the algorithm employs a grid-splitting method to map each instance to its respective grid while filtering out the grids with lower densities. Subsequently, for each instance e_i within the retained grids, it executes the related step of local density, relative distance, and fuzzy membership degree calculation to ultimately generate all the maximal fuzzy clusters.

Grid-splitting: First, the width $\sqrt{2}d$ grid is used to split the global space, where d is a user-specified neighbor distance threshold (this threshold will also be used in the pattern mining phase). Subsequently, spatial instances are mapped to the grids. Using the hash table's key value, we store the respective instance in the corresponding grid. Thus, the list of instances in each grid can be easily accessed by the hash tables, which facilitates the subsequent calculation of the number of instances included in each grid. The grid-splitting method is shown in Algorithm 1.

Algorithm 1: Grid-splitting (I,$\sqrt{2}d$)

Input: I: a set of spatial instances; $\sqrt{2}d$: the grid width.
Output: grids: a set of grids containing all instances.

Steps:
// The position of minimum abscissa
1. $min\ x \leftarrow \min\{e_1.x, e_2.x, \ldots, e_{i-1}.x, e_i.x\}$;
// The position of minimum ordinate
2. $min\ y \leftarrow \min\{e_1.y, e_2.y, \ldots, e_{i-1}.y, e_i.y\}$;
3. grids ← *InitHashtable*(); // Initialize the hash table
4. **For each** *instance* $e_i \in I$ **do** // Map instances to grids
5. $i \leftarrow \lceil (e_i.x - min\ x)/\sqrt{2}d \rceil$;
6. $j \leftarrow \lceil (e_i.y - min\ y)/\sqrt{2}d \rceil$;
7. grids$[i,j] \cup \{e_i\}$; // Incorporate the instance into the corresponding grid
8. **Return** grids; // A set of grids containing all instances
End Function

Steps 1–8 are to map spatial instances to the grids. Steps 1–3 are to create a hash table for the grids. Subsequently, in steps 4–8, the instances are mapped to the respective grid using the build hash table. Figure 1(a) illustrates a set of spatial instances under the grid-splitting.

After the grid-splitting, despite the ability to swiftly locate instances within each grid using hash tables, it is still necessary to iterate through all instances within each grid for local density calculations. As a result, the **grid-filtering** can serve as a further optimization to filter out grids with fewer instances. This aids in the rapid identification of cluster centers and reduces the time consumption for local density computation.

Density of grids: given a collection I of instances in the space and the space is divided into $\{s_1, s_2, \ldots, s_i\}$ grids according to the grid-splitting approach, the corresponding grid number is $\{G_{s_1}, G_{s_2}, \ldots, G_{s_i}\}$, the density of G_{s_i} is defined as follows:

$$\rho_{s_i} = count(G_{s_i}) \tag{6}$$

where $count()$ denotes the number of instances in G_{s_i}.

The instances are mapped into corresponding grids by using gird-splitting. Firstly, it needs to calculate the density of the grid G_{s_i}. Subsequently, we sort the grid densities calculated by the Equation (6) in descending order and set the screening ratio a. We first select the 'dense' grids with density in the top a%, and remove the other grids. Focus on the instances in the screened grids to find the cluster centers quickly. In this study, due to the large scale and tight distribution of datasets, we set $a = 50\%$, that is the instances in the top 50% "dense" grids are selected. Ultimately, the cluster centers can be identified quickly by filtering out the grids with lower densities. As shown in Figure 1(b), we remove grids with a density below 2 to obtain the set of instances after the grid-filtering.

Efficient density peak clustering (EDPC) algorithm: Based on the above methods, the EDPC algorithm is obtained. The EDPC not only discovers regions with ambiguous boundaries, but also greatly reduces the time consumption for local density calculation. The EDPC algorithm for generating maximal fuzzy clusters is shown in Algorithm 2.

Algorithm 2: EDPC (I, $\sqrt{2}d$, k, η)

Input: I: a set of spatial instances; $\sqrt{2}d$: the grid width; k: the number of k-nearest neighbors; η: the threshold of fuzzy membership degree.
Output: MF_c (a set of maximal fuzzy clusters).

Steps:
1. $CC \leftarrow \varnothing$, $\mathrm{MF_c} \leftarrow \varnothing$, $\mathrm{G_r} \leftarrow \varnothing$, $\mathrm{FS_c} \leftarrow \varnothing$; // Initialize variables
2. $\mathrm{grids} = Grid-splitting\ (I,\ \sqrt{2}d)$;// Grid-splitting
3. $G_r \leftarrow$ select grids with desity in top 50%; //Grid-filtering
4. **For each** $e_i \in G_r$ **do**
5. $KNN(e_i) = KNN\ Distance\ Calculate(e_i, k)$;
6. $\rho(e_i) = Local\ Density\ Calculate(e_i,\ KNN(e_i), d_1, d_2)$;
7. $\delta(e_i) = Relative\ Distance\ Calculate(e_i)$;
8. **For each** $e_i \in G_r$ **do** // Select cluster center
9. **If** $\delta(e_i) \geq d_2$ **then**
10. $CC \cup \{e_i\}$;
11. **For each** $c_i \in CC$ **do** //Calculate the fuzzy membership degree
12. **For each** $e_j \in$ grids **do**
13. $u_{ij}(e_j, c_i) = Fuzzy\ Membership\ Calculate(e_j, c_i)$;
14. **If** $u_{ij}(e_j, c_i) \geq \eta$ **then**
15. $F_c \cup \{e_j\}$; // Combine non-cluster centers to fuzzy clusters // $\{F_c \cup \{c_i\}\}$ denotes a fuzzy cluster whose cluster center is c_i
16. $FS_c \leftarrow FS_c \cup \{F_c \cup \{c_i\}\}$;//Combine all fuzzy clusters into FS_c
17. **For each** $F_c \subseteq FS_c$ **do**
18. **If** $F_c \nsubseteq$ *any other fuzzy clusters* **then**
19. $MF_c \cup F_c$;
20. **Return** MF_c; // A set of maximal fuzzy clusters
End Function

Step 1 initiate the sets for stored cluster centers (CC), screened grids (G_r), fuzzy clusters (FS_c), and maximal fuzzy clusters (MF_c). Steps 2–3 generate the screened grids G_r by calling Algorithm 1 and grid-filtering. Steps 4–7 compute the k-nearest neighbor distances of each instance e_i within the grids G_r. Subsequently, calculate the local density based on its respective k-nearest neighbor distances and store these values in the set ρ. Next, search for the nearest instance with a density greater than that of e_i and calculate the minimum distance to that instance as a relative distance to be stored in δ. Steps 8–10 traverse through the relative distance values of all instances within G_r. If the relative distance of an instance surpasses the predefined threshold d_2, designate it as a cluster center and add it to the set CC. Steps 11–16 calculate the fuzzy membership degree of each instance e_j to cluster center instance c_i by traversing the set of spatial instances. Instances e_j with the fuzzy membership degree exceeding a specified threshold value η are added to the fuzzy cluster corresponding to cluster center c_i. Steps 17–20 return the set of all fuzzy clusters by traversing the set of all fuzzy clusters FS_c. If a fuzzy cluster is not a subset of any other fuzzy cluster, incorporate it into the set of maximal fuzzy clusters MF_c. Finally, return the set of all maximal fuzzy clusters.

In addition, we implemented a multi-threaded **parallel approach** in the clustering process, which utilizes greedy combination and pre-allocation strategy within a thread pool, and tasks are handled by either immediate execution or queuing for later execution, ensuring that thread resources are allocated efficiently through the utilization of a thread pool, thus improving the efficiency of the clustering process.

The EDPC algorithm efficiently identifies cluster centers based on the grid-splitting and the grid-filtering. It allocates non-cluster center points to clusters based on fuzzy theory, ultimately achieving a soft division of regions across the study area. From the above discussion, we can conclude that with the increasing number of instances, more and more grids are filtered out, which significantly improves the computational efficiency of the EDPC algorithm.

2.2. Maximal fuzzy grid cliques (MFGC)

After the regional division, the following step is to conduct regional co-location pattern (RCP) mining in each region. Traditional RCP mining methods utilize the **participation ratio** (PR) and **participation index** (PI) metrics to measure the prevalence of patterns in regions, where the PR characterizes the proportion of a feature's instances in a RCP C occur in the participating instances of C, and the PI denotes the minimum value of PRs for all features in C. The **participating instances** of C refer to instances that support the prevalence of C.

The limitation of the traditional mining methods based on the PI metric is that it only focuses on the count of participating instances in patterns and ignore the fuzzy neighbor relationship between instances. This often leads to an overestimation of pattern prevalence. Taking inspiration from Hu et al. [10], we introduce the maximal fuzzy grid cliques (MFGCs) for mining FRCPs to search for participating instances of patterns in local regions, which solves the issue of overestimation of pattern prevalence in traditional RCP mining. Since FRCP mining is performed based on MFGCs, we first give the relevant concepts of pattern prevalence calculation using MFGCs, and then give the framework of FRCP mining.

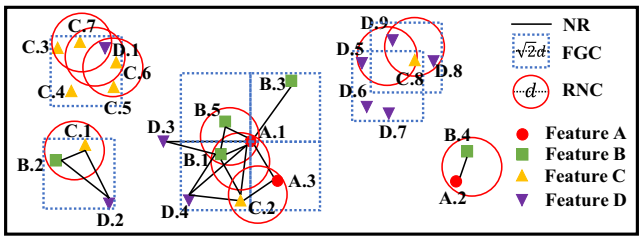
Fuzzy neighbor degree: given instances $e_i \in I$ and $e_i' \in I$, the fuzzy neighbor degree (FND) between e_i and e_i', satisfying $e_i.f \neq e_i'.f$, is defined as:

$$FND(e_i, e_i') = \begin{Bmatrix} 1 & \Delta \leq d \\ 1 - \frac{\Delta - d}{d \times (\sqrt{2}-1)} & d < \Delta \leq \sqrt{2}d \\ 0 & \Delta > \sqrt{2}d \end{Bmatrix}, \quad (7)$$

where Δ is the distance between the two instances, denoted as $\Delta = max\{|\ e_i.x - e_i'.x\ |, |\ e_i.y - e_i'.y\ |\}$, two instances are said to satisfy the R-neighbor clique (RNC) if the Δ between them is less than or equal to d ($d < \sqrt{2}d$), $\sqrt{2}d$ is the grid width by spatial grid splitting, x and y denote the abscissa and ordinate of e_i respectively.

Firstly, we can derive the **fuzzy grid neighbor relationship** (FGNR) between instances according to Equation (7), i.e., the instances e_i and e_i' are said to satisfy the FGNR, if $FND(e_i, e_i') > 0$. As shown in Figure 2, the neighbor relationships between instances are denoted by NR. For example, {A.1, B.1} is an RNC and satisfies FGNR. Next, we can obtain the **fuzzy grid neighbor** (FGN) of instance e_i based on FGNR. The FGN(e_i) of an instance e_i is composed of e_i and a set of instances that satisfy FGNR from e_i and are different from the feature of e_i. For instance, FGN(A.1) = {A.1, B.1, B.3, B.5, C.2, D.4}. The FGN of instance e_i with feature f is denoted as FGN(e_i, f), e.g., FGN(A.1, B) = {B.1, B.3, B.5}.

Figure 2
A spatial dataset with neighbor relationships



We can measure the fuzzy neighbor relationship between instances by utilizing FND and in this way generate MFGCs for the subsequent search for participating instances of patterns in regions.

Maximal fuzzy grid clique: a FGC is a subset I_1 of spatial instance set I, all of whose instances form a clique under the FGNR. If I_1 is not a subset of any other FGCs, we call I_1 as a maximal fuzzy grid clique (MFGC).

Drawing on the MFGCs generation approach described in Hu et al. [10], for instances e_i and e_i' within grids, four regions (r_1, r_2, r_3, r_4) are generated by coordinate translation to generate the FGC of e_i. For region r_1, we add e_i' to the set of FGCs for e_i if the distance between e_i and e_i' satisfies $e_i.x - e_i'.x \leq \sqrt{2}d$ and $e_i'.y - e_i.y \leq \sqrt{2}d$, and the FGCs of other regions are generated similarly. Subsequently, it performs a de-duplication operation on all FGCs by removing the cliques that are already included in other FGCs and finally generates a set of all MFGCs. In addition, if a FGC of an instance e_i contained in a candidate pattern C has already been found in region r_1, the search is halted in regions r_2, r_3, and r_4, avoiding redundant searches for FGC generation. As shown in Figure 2, {B.2, C.1, D.2} is a FGC. {C.3, C.4, C.5, C.6, C.7, D.1} is a MFGC.

After obtaining MFGCs, we can subsequently search for participating instances of patterns in regions utilizing MFGCs.

Fuzzy participating instance: for a feature $f \in C$ where C is a FRCP, $\forall$ instance e_i with feature f, if there exists a MFGC containing pattern C that contains e_i, we refer to e_i as a fuzzy participating instance (FPI) of C. The set of all fuzzy participating instances of feature f in pattern C is denoted as $FPI(C, f)$.

For example, C = {B, C, D} in Figure 2, the FPI ({B, C, D}, B) = {B.1, B.2}. Because all these instances of feature B are contained in the MFGCs containing pattern {B, C, D}.

Accumulated instance score: Firstly, we utilize the FND between e_i and its FGNs to calculate the instance score of e_i, is defined as:

$$S(e_i) = \frac{\sum_{e_i' \in FGN(e_i)} (FND(e_i, e_i') * FND(e_i, e_i'))}{\sum_{e_i' \in FGN(e_i)} FND(e_i, e_i')} \tag{8}$$

$S(e_i)$ reflects the relevance between instance e_i and its neighbors. A higher $S(e_i)$ means that e_i is more relevant to its neighbors and contributes more to them. For instance, FGN(B.2) = {B.2, C.1, D.2}, FND(B.2, C.1) = 1, FND(B.2, D.2) = 0.7, and S(B.2) is calculated as follows. $S(B.2) = \frac{FND(\text{B.2,B.2})^2 + FND(\text{B.2,C.1})^2 + FND(\text{B.2,D.2})^2}{FND(\text{B.2,B.2}) + FND(\text{B.2,C.1}) + FND(\text{B.2,D.2})} = \frac{1^2+1^2+0.7^2}{1+1+0.7} = 0.92$. In the same way, $S(A.1) = \frac{1^2+1^2+0.6^2+1^2+0.9^2+0.4^2}{1+1+0.6+1+0.9+0.4} = 0.88$. S(A.1) < S(B.2) indicates that B.2 contributes more to its neighbors than A.1 contributes to its neighbors.

Next, to better reflect the influence of FGN(e_i) on e_i, we utilize the **sharing instance score** (SIS) based on $S(e_i)$ to calculate the influence of features in FGN(e_i) on e_i, which is defined as follows. Given an instance e_i, for $e_i' \in FGN(e_i)$ and its feature $f_1 = e_i'.f$, the SIS of the instance e_i that received by e_i' is denoted as $SIS(e_i, e_i')$, is defined as:

$$SIS(e_i, e_i') = \frac{S(e_i)}{|FGN(e_i, f_1)|} \tag{9}$$

From Equation (9), its denominator is the number of instances with feature f_1 in the FGN of e_i. If an instance e_i is shared by multiple instances of the same feature, the instance score of e_i will be distributed to the neighboring instances with the same feature on average. As shown in Figure 2, FGN(A.1) = {A.1, B.1, B.3, B.5, C.2, D.4}, instance A.1 is shared by 3 instances with feature B, So the $SIS(A.1, \text{B.1}) = SIS(A.1, \text{B.3}) = SIS(A.1, \text{B.5}) = \frac{S(A.1)}{3} = \frac{0.88}{3} = 0.29$, on the contrary, $SIS(B.2, D.2) = S(B.2) = 0.92$, since no other instances with feature D shares B.2 with instance D.2.

Then, to better calculate the contribution of the participating instances in patterns, we need to accumulate the SIS obtained by instance e_i from its neighboring features type $f_1, f_1 \neq e_i.f$, i.e., **accumulated instance score** of e_i. It is defined as shown in Equation (10).

$$AIS(e_i, f_1) = min\left\{ \sum_{e_i' \in FGN(e_i), f_1 = e_i'.f} \text{SIS}(e_i, e_i'), 1 \right\}. \tag{10}$$

We limit the maximum accumulated instance score to 1. For example, $AS(A.1, B) = min\{\sum(0.29 + 0.29 + 0.29), 1\} = 0.88$, $AIS(B.2, D) = min\{0.92,\ 1\} = 0.92$.

Fuzzy participating contribution index (FPCI): The FPCI denotes the minimum value of the fuzzy participating contribution ratio (FPCR) for all features in a pattern, while the FPCR is defined as follows. Given an instance $e_i \in FPI(C, f)$, for feature $f \in C, f_1 \in C - \{f\}$, where C is an FRCP, the FPCR of f to C is defined as:

$$FPCR(C, f) = \frac{1}{|I(f)|} * \sum_{e_i \in FPI(C, f)} \left(\sum_{f_1 \in C - \{f\}} \frac{AIS(e_i, f_1)}{|C - \{f\}|} \right) \tag{11}$$

From Equation (11), we derive that the $FPCR(C, f)$ indicates the contribution of f in C, where $|I(f)|$ denotes the number of instances with feature f in C, and $|C - \{f\}|$ denotes the total number of instances with other features in C. A higher $FPCR(C, f)$ means that C contains more participating instances of feature f. Based on the above methods of measuring pattern prevalence, we can ultimately derive all the prevalent FRCPs in regions.

Lemma 1. The anti-monotone property. The FPCR and the FPCI monotonically non-increase as the size of FRCPs increases.

Proof. Suppose two FRCPs C_1 and C_2, where $C_1 \subseteq C_2$ and $|C_1|2$. From concept of FPI, it follows that the features contained in a FRCP and the FGNR between the corresponding instances of the features will influence its FPI. Given a feature $f \in C_1$ and $\in C_2$, if an instance e_i of the feature f is a participating instance in C_2, then it must also be a participating instance in C_1. That is, $FPI(C_2, f) \subseteq FPI(C_1, f)$. In addition, according to the concept of FPCR, we have $\sum_{e_i \in FPI(C_2, f)} \left(\sum_{f_t \in C - \{f\}} \frac{AIS(e_i, t)}{|C_2 - \{f\}|} \right) \leq \sum_{e_i \in FPI(C_1, f)} \left(\sum_{f_t \in C - \{f\}} \frac{AIS(e_i, t)}{|C_1 - \{f\}|} \right)$, so there is $FPCR(C_2, f) \leq FPCR(C_1, f)$. Consequently, the FPCR is anti-monotone. The FPCI is also anti-monotone because it is the minimum value of the FPCR for all features in the FRCP, i.e., $FPCI(C_2) = FPCI(C_1 \cup f_{k+1}) = min_{f \in C_1 \cup f_{k+1}}\{FPCR(C_1 \cup f_{k+1}, f)\} \leq min_{f \in C_1}\{FPCR(C_1 \cup f_{k+1}, f)\} \leq min_{f \in C_1}\{FPCR(C_1, f)\} = FPCI(C_1)$.

Now we summarize the entire FRCP mining (FRCPM) framework with the pseudocode presented in Algorithm 3.

Algorithm 3: FRCPM-EDPC-MFGC(I, $\sqrt{2}d$, k, η, p)

Input: I: a set of spatial instances; $\sqrt{2}d$: the grid width; k: the number of k-nearest neighbors; η: the threshold of fuzzy membership degree; p: the prevalence threshold.
Output: *PFRCP* (a set of prevalent FRCPs).

Steps:
1. $MF_c = EDPC\ (I, \sqrt{2}d,\ k,\ \eta)$; //maximal fuzzy cluster set
2. **For** $MF \in MF_c$ *do* // FRCP mining in each MF
3. $C_h \leftarrow C_2$, $P_h \leftarrow P_2$, $FPRCP \leftarrow \varnothing$, h $\leftarrow 2$ // Initialize variables
4. **For** $I \subseteq MF$ **do** // The grid splitting for instances within MF
5. grids $= grid - splitting\ (I, \sqrt{2}d)$;
6. $T_s = MFGC\ generation$ (grids);// Generate MFGCs
7. **While** $P_h \neq \varnothing$ **do**
8. $C_h \leftarrow CandidatePatternGeneration(P_{h-1})$;
9. **If** $C_h \neq \varnothing$ **then**
10. **For each** *Candidate pattern* $C \in C_h$
11. **For each** *transaction* $T \in T_s$ **do**
12. **If** T contains all the features in C **then**
13. **For each** *instance* $_i \in T$ **do**
14. **If** $e_i.f \in C \cap e_i \notin FPI(C, f)$ **then**
15. $FPI(C, f) \cup \{e_i\}$;
16. **If** $FPCI(C,f)p$ **do**
17. $P_h \cup \{C\}$;
18. $Remove(T_s, h)$;
19. $PFRCP \leftarrow PFRCP \cup \{P_h\}$;
20. $h \leftarrow h + 1$
21. **Return** *PFRCP*; // Return the set of prevalent FRCPs
End Function

In Algorithm 3, Step 1 generates a set of maximal fuzzy cluster MF_c by the EDPC algorithm. Steps 2–21 are to mine FRCPs within each cluster and ultimately return the set of prevalent FRCPs within all clusters. In detail, Step 6 generates all MFGCs to represent the spatial transaction. The anti-monotone property, as described in Lemma 1, ensures that when an FRCP is in size-h FRCPs P_h, all of its subsets must also be in P_{h-1}. Consequently, Step 8 generates a candidate pattern C_h by combining any two of the h-1 size prevalent patterns. Steps 9–15 identify the participating instances of C based on MFGCs. Steps 16–17 calculate the fuzzy participating contribution index (FPCI) for pattern C and check whether the pattern is prevalent. Step 18 evaluates and removes transactions that contain fewer than $h + 1$ features or instances to optimize memory usage and scanning efficiency. In Step 19, all h-size prevalent FRCPs are collected. The process of mining prevalent FRCPs is finished by constantly increasing the size of h until the prevalent pattern P_h does not exist. Finally, Step 21 returns the mining results of all prevalent FRCPs.

3. Complexity Analysis

This section provides a detailed analysis of the time and space complexity associated with the FRCPM-EDPC-MFGC algorithm. Let's denote the number of spatial instances as n. After the grid-filtering, we select a subset of spatial instances, which contains the number of instances denoted as n_g ($n_g < n$), and the number of maximal fuzzy clusters is represented by M_F. The number of instances in each maximal fuzzy cluster is denoted as n_f ($n_f < n$). The number of nearest neighbors of an instance e_i is denoted as k, and the size of patterns is indicated by h. Additionally, during the process of generating MFGCs, the total number of instances in the grids where instance e_i is located, as well as in its neighboring grids is N_{e_i}. The total number of generated FGCs of instance e_i is denoted as N_F, and the number of FRCPs mined by FRCPM-EDPC-MFGC algorithm is represented by $|R|$.

The time complexity of mapping the data points into the grids is $O(n)$. Selecting the grids with higher density using fast sorting is $O(nlogn)$. Computing the k-nearest neighbors is $O(n_g^2)$, while the calculation of local density is $O(k * n_g)$. The computation of relative distance is $O(n_g^2)$, and the selection of cluster center is $O(n_g)$. The time complexity computing the fuzzy membership degree is $O((n - M_F) * M_F^2)$. The generation of maximal fuzzy clusters requires $O((n - M_F) * M_F^2)$ operations. The time complexity of searching for FGCs contained in the four regions (r_1, r_2, r_3, r_4) of each instance e_i within each cluster is $O(n_f * N_{e_i})$. The step of generating MFGCs by removing redundant FGCs, which have a time complexity of $O(N_F^2)$. As shown in Algorithm 3, we generate a candidate pattern C_h by combining any two of the h-1 size prevalent patterns P_{h-1} (Step 8), which has a time complexity of $O(|P_{h-1}|^2)$, where $|P_{h-1}|$ denotes the number of h-1 size prevalent FRCPs. Steps 9–17 search for participating instances of each feature f in candidate FRCP C by traversing the transaction represented by MFGCs, and thus checking whether a size-h candidate FRCP is prevalent, with a time complexity of $O(|C_h| * |T_s|)$, where $|C_h|$ indicates the number of candidates FRCPs with size-h, $|T_s|$ represents the number of transactions. We then compute the FPCI of patterns by collecting the set of participating instances for each feature in C, which has a time complexity of $O(|FPI| * (|C| - 1))$, where $|FPI|$ is the total number of participating instances in C. In conclusion, the final time complexity of the FRCPM-EDPC-MFGC algorithm is $O(n + nlogn + n_g^2 + k * n_g + (n - M_F) * M_F^2 + M_F * (n_f * N_{e_i} + N_F^2 + h * (|P_{h-1}|^2 + |C_h| * |T_s| + |FPI| * (|C| - 1))))$.

The space complexity for storing k-nearest neighbors is $O(k * n_g)$, the space complexity for storing local density and relative distance is $O(n_g)$. Storing the fuzzy membership degree is $((n - M_F) * M_F)$, mining FRCPs is $O(|T_s| + |R| + 1)$. Based on the above analysis, the total space complexity of the FRCPM-EDPC-MFGC algorithm is $O(k * n_g + (n - M_F) * M_F + M_F * (|T_s| + |R| + 1))$.

4. Experimental Performance Evaluation

In this section, we primarily focus on examining the impact of several key factors, including the number of instances $|I|$, the number of features $|F|$, gird width w ($w = \sqrt{2}d$), the number of threads t, on both the runtime and the results of FRCP mining algorithm achieved by the FRCPM-EDPC-MFGC (EM) of this study and RCP mining algorithm achieved by the DPC-Joinless (DJ) of Jiang et al. [16]. We control a variable and fix other parameters in each experiment, as shown in Table 1, where *min_prev* is the minimum prevalence threshold. Specifically, we conduct experiments utilizing the synthetic dataset obeying the Poisson distribution in Section 4.1, as well as the cluster-distributed Shenzhen POI dataset with 71606 instances and 13 features in Section 4.2, where the meanings of features and the corresponding number of instances as well as the distribution in the

Table 1
Parameter settings corresponding to different variables

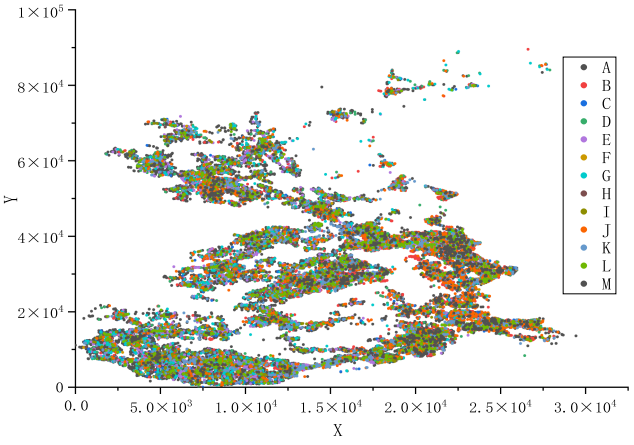
Variables	Parameter settings
$\|I\|$	$\|F\| = 10$, $w = 500\sqrt{2}$, $min\_prev = 0.37$, $t = 8$.
$\|F\|$	$\|I\| = 20000$, $w = 500\sqrt{2}$, $min\_prev = 0.23$, $t = 8$.
w	$\|I\| = 20000$, $\|F\| = 10$, $min\_prev = 0.3$, $t = 8$.
t	$\|I\| = 50000$, $\|F\| = 10$, $w = 500\sqrt{2}$, $min\_prev = 0.48$.

Shenzhen POI dataset are shown in Table 2 and Figure 3, respectively. The configuration of the computer used for the following experiments is Intel(R) Core(TM) i7-6700 with 4 cores and 8 threads as well as 16 GB of RAM.

Table 2
Details of Shenzhen POI dataset

Feature	Meaning	Instances	Feature	Meaning	Instances
A	Company	20230	B	Residence	8955
C	Snack bar	6666	D	Supermarket	6300
E	Furniture store	5864	F	Clothing store	5513
G	Restaurant	4538	H	Barber shop	3888
I	Electronic store	2447	J	Parking lot	2393
K	Post office	1761	L	Farmers market	1603
M	School	1448			

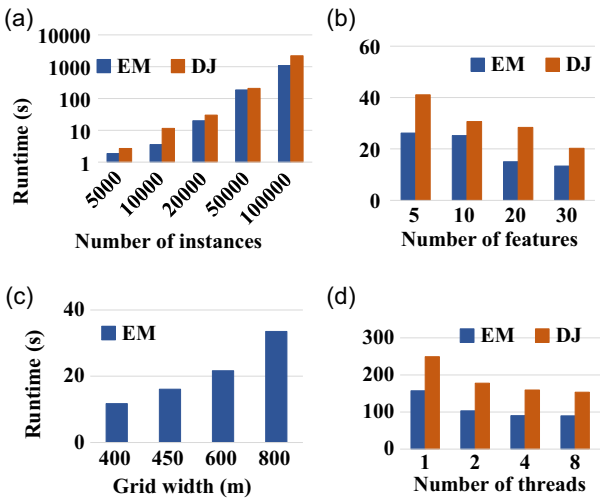
Figure 3
Distribution of Shenzhen POI dataset



4.1. Effect of the parameters on the execution time

It is evident from Figure 4(a) that the algorithm's runtime progressively increases with an increase in the number of instances. This can be attributed to the fact that the number of instances has a direct impact on the number of fuzzy clusters, which in turn affects the number of neighbor relationships between instances. Consequently, it leads the algorithm to process a larger number of instances and engage in more intricate calculations, resulting in longer execution times. The EM algorithm uses grid-filtering to reduce the clustering time and uses MFGCs to find the participating

Figure 4
Effect of parameters for execution time



instances of patterns, while the DJ algorithm needs to calculate the neighbor relationships between all instances. Therefore, the EM algorithm is faster than the DJ algorithm.

Concerning the number of features, there are two key aspects to consider. Firstly, with the same number of instances, an increase in the number of features results in a decrease in the number of instances within each feature, while the proximity between the features intensifies. This leads to a greater number of neighbor relationships, demanding more time for the algorithms to detect these relationships. Secondly, this increase in the number of features changes the instance distribution, affecting not only the number of fuzzy clusters but also the neighbor relationships within these clusters. The interplay of these two factors may cause the runtime as shown in Figure 4(b): decreasing progressively as the number of features increases. The DJ has an overall longer runtime compared to the EM because it needs to identify more neighbor relationships between different features in this context, while the EM utilizes MFGC to search for participating instances, resulting in relatively lower time consumption.

Concerning the grid width w, on the one hand, according to Equation (6), w has an impact on the density of grids, which subsequently influences the number of fuzzy clusters. In addition, increasing w causes the EM to calculate more neighbor relationships. Due to the combined effect of these two aspects, the runtime steadily increases with increasing w, as shown in Figure 4(c).

Concerning the comparison of the runtime between the algorithms and its parallel version, the results reveal that the parallel EM algorithm exhibits a lower overall time cost compared to the DJ algorithm. Moreover, some tasks with longer execution times may be blocked due to the greed strategy and predetermined allocation of threads within the thread pool, increasing the overall time cost of the algorithm. For instance, as shown in Figure 4(d), increasing the number of threads from 4 to 8 does not lead to a substantial reduction in the algorithm's runtime (but still has the advantage of time).

Finally, we compute the ratio of the difference between the runtime of the EM and the DJ to the total runtime of the DJ and then calculate the mean value of this ratio for all parameters. We can conclude that compared to the DJ algorithm, the EM algorithm saves about 40% of the time cost. In summary, the EM algorithm shows higher performing efficiency.

4.2. Analysis on the real dataset

In this section, we applied the EM algorithm to the Shenzhen POI dataset with the following parameters: $k = 150$, $t = 8$, $w = 220\sqrt{2}$, $min\_prev = 0.4$. We conducted a comparative analysis of the mining results with both the global mining algorithm GCP [10] and the DJ algorithm.

4.2.1. Comparison with the GCP mining algorithm

Table 3 describes the runtime of each algorithm and the mean values of the neighboring distances between instances corresponding to the features within some of the mined patterns. We can observe that the EM algorithm mines patterns that cannot be mined in the GCP mining method, and we select the pattern {A, J} to elaborate. As can be seen in Table 2, the number of instances of feature A is relatively large and the number of instances of feature J is relatively small. Although the GCP algorithm also considers fuzzy neighbor relationships between instances, it cannot discover the pattern {A, J} that occurs prevalently only in a local region.

Table 3
Details of comparison with the mining results for different algorithms

Algorithm		EM	DJ	GCP
Runtime(s)		513.783	581.647	177.812
Mean of distance	{F, I}	3905.528	4200.954	22319.901
	{I, K}	4980.47	5126.805	×
	{A, B}	4990.716	5780.569	20081.509
	{C, G}	6069.569	5411.741	22470.22
	{C, D}	6866.761	4476.587	22336.801
	{D, E}	6534.921	5819.532	22963.611
	{A, J}	4418.74	5310.059	×

Note: * "×" indicates that the pattern was not mined.

In addition, we can see from Table 3 that the distribution of patterns mined by the EM algorithm is more compact, i.e., the relative distances between the instances of different features are closer compared to the GCP mining method. This can be confirmed by the mean values of the distances between instances corresponding to the features in these patterns. In summary, the local region mining algorithm EM is better than the global mining algorithm GCP.

4.2.2. Comparison with the other RCP mining algorithm

In this section, we present a comparative analysis of the EM and DJ algorithms. To reflect the difference in the prevalence of the FPCI and the PI metric, we analyze the mining results by examining top-15 patterns ranked in descending order of prevalence. Table 4 presents the top-15 prevalent patterns discovered using the FPCI metric, along with the corresponding prevalence values and rankings obtained using the PI metric. In particular, we focus our attention on the top-2 pattern in Table 4.

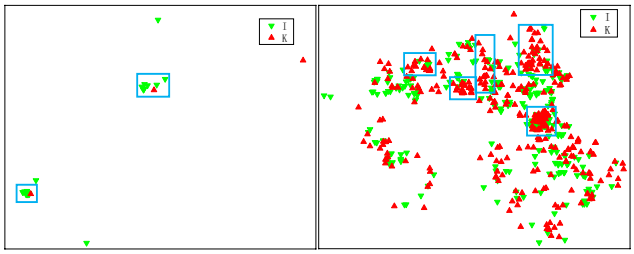
We first compare the differences in the distribution of instances with different features in the prevalent region of FRCP {I, K} mined by EM and that of RCP {I, K} mined by DJ respectively. From Table 4, we can see that pattern {I, K} ranks 2nd in the FPCI metric, while it ranks 68th in the PI metric. This is because we can observe from the left side of Figure 5 that the number of instances with features I and K in the pattern {I, K} mined by EM is fewer, but the multiple instances of feature I tend to cluster

Table 4
Mining results of patterns with top-15 prevalence

Top-15 patterns	Prevalence measure			
	FPCI	Rank	PI	Rank
{F, I}	0.57713	1	0.77273	18
{I, K}	0.56112	2	0.50992	68
{A, B}	0.50465	3	0.90633	1
{C, G}	0.50432	4	0.87635	4
{C, D}	0.49495	5	0.88853	2
...	...	...	...	...
{D, E}	0.47303	14	0.77968	16
{A, J}	0.47118	15	0.82692	8

Note: * "×" indicates that the pattern was not mined.

Figure 5
Distribution of instances in the prevalent regions of pattern {I, K} mined by the EM (left) and DJ (right)



around feature K in local regions, leading to the complex neighbor relationship between instances of feature I and K. The FPCI metric considers the contribution of participating instances in a pattern by incorporating the fuzzy neighbor degree and sharing instance score as a way to mine the pattern {I, K}.

In contrast, as evident from the right side of Figure 5, the instances of different features in the prevalent region of RCP {I, K} identified by the DJ algorithm are relatively distant from each other, as instances tend to gather within the same feature, resulting in a more dispersed distribution among different features. The PI metric solely considers the number of instances involved in the pattern without acknowledging the sharing neighbor relationships among instances, resulting in a lower PI value for the pattern {I, K} mined by the DJ algorithm compared to the other patterns mined by the DJ. Meanwhile, Table 3 indicates that the total runtime of the EM algorithm is 513.783 s, which is lower than that of the DJ algorithm with 581.647 s. The mean value of the distances between instances within the pattern {I, K} mined by EM is 4980.47 m, which is smaller than that of DJ with 5126.805 m.

Through a comprehensive analysis of the above experimental results, it can be concluded that the EM algorithm excels in performing efficiency and identifying patterns with a more tightly distributed set of instances.

5. Conclusion and Future Work

In this study, we propose a novel FRCP mining algorithm that combines the efficient density peak clustering (EDPC) and the maximal fuzzy grid clusters (MFGCs). Firstly, the EDPC utilizes grid-filtering to filter out grids with lower densities, which effectively solves the problem of excessive time consumption for calculating the local density and then generates local regions with

ambiguous boundaries. Secondly, a set of MFGCs is generated based on fuzzy neighbor relationships within each local region to collect the participating instances of patterns, and the FPCI metric is used to measure pattern prevalence, which effectively solves the problem of overestimation of pattern prevalence in traditional RCP mining. Experimental results demonstrate that the proposed FRCPM-EDPC-MFGC algorithm surpasses not only the global mining algorithm GCP but also the current state-of-the-art RCP mining algorithm DPC-Joinless. Moreover, it successfully identifies prevalent regions with tighter instance distributions.

However, this study simplifies the representation of physical entities as point instances, exploiting a simple greedy strategy to allocate threads in the parallelization process. Additionally, determining thresholds (e.g., grid width) is also a challenge. Therefore, the further research directions are as follows: (1) expanding the research scope to various types of spatial instances, such as those with uncertain shape, (2) considering more efficient parallelization algorithms applied in FRCP mining within each maximal fuzzy cluster, and (3) improving the adaptability of grid width and prevalence threshold computation to minimize user intervention.

Funding Support

This work is supported by the National Natural Science Foundation of China (62276227, 62266050, 62306266), the Project of Innovative Research Team of Yunnan Province (2018HC019), and the Yunnan Fundamental Research Projects (202201AS070015).

Ethical Statement

This study does not contain any studies with human or animal subjects performed by any of the authors.

Conflicts of Interest

Lizhen Wang is an Editorial Board Member for *Journal of Data Science and Intelligent Systems* and was not involved in the editorial review or the decision to publish this article. The authors declare that they have no conflicts of interest to this work.

Data Availability Statement

The data that support the findings of this study are openly available in GitHub at https://github.com/Vansank/DataSet.git.

Author Contribution Statement

Tao Zhou: Conceptualization, Methodology, Software, Formal analysis, Investigation, Data curation, Writing – original draft, Visualization. **Lizhen Wang:** Conceptualization, Methodology, Validation, Formal analysis, Resources, Writing – review & editing, Supervision, Project administration, Funding acquisition. **Dongsheng Wang:** Investigation, Visualization. **Vanha Tran:** Validation, Data curation, Writing – review & editing.

References

[1] Luo, N., Yu, H., You, Z., Li, Y., Zhou, T., Jiao, Y., . . . , & Qiao, S. (2023). Fuzzy logic and neural network-based risk assessment model for import and export enterprises: A review. *Journal of Data Science and Intelligent Systems*, *1*(1), 2–11. https://doi.org/10.47852/bonviewJDSIS32021078

[2] Li, L., Cheng, J., Bannister, J., & Mai, X. (2022). Geographically and temporally weighted co-location quotient: An analysis of spatiotemporal crime patterns in greater Manchester. *International Journal of Geographical Information Science*, *36*(5), 918–942. https://doi.org/10.1080/13658816.2022.2029454

[3] Qiu, P., Gong, Y., Zhao, Y., Cao, L., Zhang, C., & Dong, X. (2023). An efficient method for modeling nonoccurring behaviors by negative sequential patterns with loose constraints. *IEEE Transactions on Neural Networks and Learning Systems*, *34*(4), 1864–1878. https://doi.org/10.1109/TNNLS.2021.3063162

[4] Maheswari, K. G., Siva, C., & Nalinipriya, G. (2023). Optimal cluster based feature selection for intrusion detection system in web and cloud computing environment using hybrid teacher learning optimization enables deep recurrent neural network. *Computer Communications*, *202,* 145–153. https://doi.org/10.1016/j.comcom.2023.02.003

[5] Shekhar, S., & Huang, Y. (2001). Discovering spatial co-location patterns: A summary of results. In *Advances in Spatial and Temporal Databases: 7th International Symposium*, 236–256. https://doi.org/10.1007/3-540-47724-1_13

[6] Huang, Y., Shekhar, S., & Xiong, H. (2004). Discovering colocation patterns from spatial data sets: A general approach. *IEEE Transactions on Knowledge and Data Engineering*, *16*(12), 1472–1485. https://doi.org/10.1109/TKDE.2004.90

[7] Yoo, J. S., Shekhar, S., Smith, J., & Kumquat, J. P. (2004). A partial join approach for mining co-location patterns. In *Proceedings of the 12th Annual ACM International Workshop on Geographic Information Systems,* 241–249. https://doi.org/10.1145/1032222.1032258

[8] Yoo, J. S., & Shekhar, S. (2006). A joinless approach for mining spatial colocation patterns. *IEEE Transactions on Knowledge and Data Engineering*, *18*(10), 1323–1337. https://doi.org/10.1109/TKDE.2006.150

[9] Chan, H. K. H., Long, C., Yan, D., & Wong, R. C. W. (2019). Fraction-score: A new support measure for co-location pattern mining. In *IEEE 35th International Conference on Data Engineering*, 1514–1525. https://doi.org/10.1109/ICDE.2019.00136

[10] Hu, Z., Wang, L., Tran, V., & Chen, H. (2022). Efficiently mining spatial co-location patterns utilizing fuzzy grid cliques. *Information Sciences*, *592*, 361–388. https://doi.org/10.1016/j.ins.2022.01.059

[11] Huang, Y., & Zhang, P. (2006). On the relationships between clustering and spatial co-location pattern mining. In *18th IEEE International Conference on Tools with Artificial Intelligence*, 513–522. https://doi.org/10.1109/ICTAI.2006.91

[12] Rodriguez, A., & Laio, A. (2014). Clustering by fast search and find of density peaks. *Science*, *344*(6191), 1492–1496. https://doi.org/10.1126/science.1242072

[13] Liu, Y., Ma, Z., & Yu, F. (2017). Adaptive density peak clustering based on K-nearest neighbors with aggregating strategy. *Knowledge-Based Systems*, *133*, 208–220. https://doi.org/10.1016/j.knosys.2017.07.010

[14] Fan, C., Lee, S., Yang, Y., Oztekin, B., Li, Q., & Mostafavi, A. (2021). Effects of population co-location reduction on cross-county transmission risk of COVID-19 in the United States.

Applied Network Science, *6*(1), 14. https://doi.org/10.1007/s41109-021-00361-y

[15] Deng, M., Cai, J., Liu, Q., He, Z., & Tang, J. (2017). Multi-level method for discovery of regional co-location patterns. *International Journal of Geographical Information Science*, *31*(9), 1846–1870. https://doi.org/10.1080/13658816.2017.1334890

[16] Jiang, X., Wang, L., & Tran, V. (2023). Jīyú móhú mìdù fēngzhí jù lèi de qūyù tóngwèi móshì bìngxíng wājué suànfǎ [A parallel algorithm for regional co-location mining based on fuzzy density peak clustering]. *SCIENTIA SINICA Informationis*, *53*(7), 1281–1298. https://doi.org/10.1360/SSI-2022-0004

[17] Liu, Z., & Letchmunan, S. (2024). Representing uncertainty and imprecision in machine learning: A survey on belief functions. *Journal of King Saud University – Computer and Information Sciences*, *36*(1), 101904. https://doi.org/10.1016/j.jksuci.2023.101904

[18] Liu, Z. (2023). Credal-based fuzzy number data clustering. *Granular Computing*, *8*(6), 1907–1924. https://doi.org/10.1007/s41066-023-00410-0

[19] Liu, Z., & Letchmunan, S. (2024). Enhanced fuzzy clustering for incomplete instance with evidence combination. *ACM Transactions on Knowledge Discovery from Data*, *18*(3), 72. https://doi.org/10.1145/3638061

[20] Xu, X., Ding, S., & Shi, Z. (2018). An improved density peaks clustering algorithm with fast finding cluster centers. *Knowledge-Based Systems*, *158*, 65–74. https://doi.org/10.1016/j.knosys.2018.05.034

How to Cite: Zhou, T., Wang, L., Wang, D., & Tran, V. (2025). Fuzzy Regional Co-location Pattern Mining Based on Efficient Density Peak Clustering and Maximal Fuzzy Grid Cliques. *Journal of Data Science and Intelligent Systems*, *3*(3), 206–215. https://doi.org/10.47852/bonviewJDSIS42022134