Mining ϵ-Closed High Utility Co-location Patterns from Spatial Data

Vanha Tran[1], Lizhen Wang[2,3](✉), Shiyu Zhang[2], Jinpeng Zhang[2,4], and SonTung Pham[1]

[1] FPT University, Hanoi 155514, Vietnam
hatv14@fe.edu.vn, tungpshe140670@fpt.edu.vn
[2] Yunnan University, Kunming 650091, China
lzhwang@ynu.edu.cn, zsy608118@mail.ynu.edu.cn
[3] Dianchi College of Yunnan University, Kunming 650213, China
[4] Yunnan University of Finance and Economics, Kunming 650221, China
zjp@ynufe.edu.cn

Abstract. Traditional spatial prevalent co-location pattern mining is discovering groups of spatial features whose instances frequently appear together in nearby areas. However, it is unsuitable for many real-world applications where the significance of these instances must be considered. High utility co-location pattern (HUCP) mining is developed to find highly beneficial patterns by considering the importance of spatial instances. However, the mining result typically contains many HUCPs, making it difficult for users to absorb, comprehend, and apply. This work proposes a compressed representation of HUCPs, ϵ-closed HUCPs, that allow for a user-specified small tolerance of the information between a pattern and its supersets. If the information difference is not larger than the small tolerance it only needs to keep the supersets. Moreover, an efficient algorithm is developed to discover ϵ-closed HUCPs. The proposed algorithm avoids examining many unnecessary candidates; therefore, the performance of mining ϵ-closed HUCPs is significantly improved. A set of different numbers of features, numbers of instances, and distribution of both synthetic and real data sets are employed to evaluate the performance of the proposed method completely. The experimental results show that ϵ-closed balances the compression rate and the UPI error rate and gives a large pattern compression rate within a relatively small range of error rates. Moreover, the proposed algorithm is high-performance on dense and large data sets.

Keywords: Prevalent co-location pattern · High utility co-location pattern · ϵ-closed pattern

1 Introduction

As an important direction of spatial data mining, discovering spatial prevalent co-location patterns (PCPs) from spatial data sets is finding groups of spatial features whose spatial instances frequently occur with each other in nearby areas. For example, in a point of interest (POI) data set of a city, by using the PCP mining technique we find that banks, supermarkets, and bus stations often appear together within 200 m, i.e., {bank, supermarket, bus station} is a PCP. PCPs can expose relationship rules between features in spatial data sets and the rules have been applied in many fields such as public safety [10], disease control [11], agriculture [4,12], criminology [7,9], business [16], climate science [1], transportation and location-based services [20], and so on.

In PCP mining, spatial instances are treated equally, it only considers the prevalence of these instances that appear together in nearby of each other. However, in practice, each spatial instance has its importance, if the factor is not considered, PCPs may lose meaning in their practical applications. For example, the importance of a supermarket and a bank is not the same. The services covered by the supermarket will be more extensive than the bank in respect of the supply of goods and service objects. The impact on the surrounding populace of the supermarket will be greater than the bank. Therefore, the notion of high utility co-location patterns (HUCP) has been proposed [19]. In HUCPs, each spatial instance is assigned a utility value to reflect its significance or importance. In addition, the correlation of the spatial features in a pattern is also considered through the participating spatial instances of the pattern.

However, the mining result normally generates a large number of HUCPs and includes redundant information. It is difficult for users to understand, absorb and apply the mining result. Thus, it is very necessary to find a concise representation of the mining result. But it is not easy to find a good concise representation since it needs to meet three conditions at the same time: (1) the number of patterns should be as little as possible; (2) the redundancy information should be removed as much as possible; and (3) the error between the compressed and the original should be as small as possible.

This work focuses on devising a representation that satisfies the above three conditions. The main contributions of this paper are as follows:

- Propose an ϵ-closed high utility co-location pattern notion that is a concise representation of HUCPs. If there is a small amount of information difference between a HUCP and its supersets, the HUCP can be deleted, and it just only needs to remain the supersets. The number of HUCPs can be reduced significantly under losing a small amount of information.
- Users can control the number of output HUCPs through ϵ which defines the amount of information difference between a HUCP and its supersets.
- Design a neighboring instance-based mining algorithm. Different from the traditional generating-testing candidate mining algorithm, the proposed algorithm does not collect and validate every co-location instance of a pattern, it adopts an efficient query mechanism of a hash table.

The rest of this paper is organized as follows: Sect. 2 reviews related work. The formal definitions and properties of ϵ-closed HUCPs are described in Sect. 3. The neighboring instance-based mining algorithm is designed in detail in Sect. 4. Experiments on both synthetic and real data sets are shown in Sect. 5. Section 6 summarizes the paper and highlights the directions for future work.

2 Related Work

Compare with frequent itemset mining [13,14], PCP mining is more difficult, complex, and interesting since in PCPs, there are very complicated neighbor relationships between spatial instances in space. Thus, PCP mining has attracted many researchers. Join-based [8] has been known as the first algorithm for mining PCPs. It uses a join operation to collect co-location instances of each pattern. This operation becomes very expensive when dealing with long-size PCPs and/or dense spatial data sets. To avoid this disadvantage, many efficient PCPS mining algorithms have been proposed such as join-less [22], overlapping clique-based [17], clique-based [3], and candidate pattern tree [21].

How to efficiently discover PCPs from big data in limited execution time and computer resources has become a new challenge. Parallel PCP mining algorithms have been developed by executing on different platforms such as Hadoop and Map-reduce [15], the graphic processing unit (GPU) [2], and NoSQL [2].

The above algorithms focus on mining all complete and correct PCPs. However, the mining result normally contains too many redundant PCPs, this makes it difficult for users to understand, absorb and apply the mining result. Thus, mining concise representations of PCPs have been proposed such as maximal [16], closed [21], and non-redundant PCPs [18].

These mining algorithms mentioned above can be called traditional PCP mining algorithms. The traditional only considers the prevalence of spatial instances appearing together in nearby space, the importance or significance of spatial instances is ignored, this may cause the discovered PCPs to become meaningless. Therefore, HUCPs [19] that each spatial instance is assigned a utility value to reflect its importance are proposed. To discover HUCPs, a utility participation index (UPI) metric is designed to evaluate the interest of patterns. Unfortunately, UPI does not hold the downward closure property that is often used to reduce the candidate search space to improve mining efficiency.

Although HUCPs consider the importance of each instance, HUCPs have the same problem as the traditional PCPs, that is, the mining result contains too many patterns, making it difficult for users to absorb and apply. Therefore, in this work, we focus on finding a concise representation of HUCPs. First, an ϵ-closed HUCP notion, that relaxes the concise condition of the closure property by allowing a small difference ϵ in the UPI value between a HUCP and its supersets, is proposed. Second, a neighboring instance-based mining algorithm is designed. This algorithm is based on the neighboring spatial instances, i.e., it first finds all neighboring instances and stores the neighbor relationships into a compact hash table, then co-location instances of each pattern are obtained by querying on the hash table, it avoids verifying the neighbor relationship of each co-location instance in the generating-testing candidate mining mechanism.

3 Method

Given a set of spatial features $F = \{f_1, ..., f_m\}$ and a set of spatial instances $S = \{I_1, ..., I_m\}$ where $I_t = \{o_1, ..., o_q\}$ is the set of instances of feature $f_t \in F$.

Each spatial instance $o_i \in S$ is formed by a three-element vector, i.e., <feature type, instance identification, location>. We use $f(o_i) = f_t$ to represent the feature type of instance o_i is f_t. R is a user-defined neighbor relationship, if two spatial instances o_i and o_j have the neighbor relationship, they are denoted as $R(o_i, o_j)$. When R uses a distance metric (e.g., Euclidean, Manhattan, Minkowski distance metrics, and so on), if the distance between two instances o_i and o_j is not larger than a distance threshold d given by users, o_i and o_j satisfy the neighbor relationship R, i.e., $R(o_i, o_j) <=> distance(o_i, o_j) \leq d$. $NR(o_i) = \{o_j | R(o_i, o_j) \wedge f(o_i) \neq f(o_j)\}$ is the set of neighboring instances of o_i under R.

Definition 1 *(Utility of a spatial instance). The utility of a spatial instance $o_i \in S$ is a value $v \leq 0$ that is assigned to the instance to reflect its importance and expressed by the superscript on it, i.e., o_i^v. We denote the utility of a spatial instance o_i^v as $u(o_i^v) = v$.*

Definition 2 *(Utility of a spatial feature). The utility of a spatial feature $f_t \in F$ is the sum of the utilities of its instances and denote as*

$$u(f_t) = \sum_{i=1}^{q} u(o_i^v) \tag{1}$$

where q is the number of instances that belong to f_t.

For example, Fig. 1 shows an example of spatial data set with assigned instance utilities. A1^{15} is an instance of feature A and its utility value is 15. The solid line connects two instances to represent that they have a neighbor relationship. Table 1 lists the instances with their utility values of each spatial feature. Moreover, the utility of each feature also is drawn in Table 1.

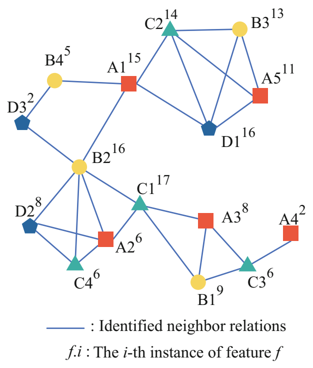


Fig. 1. An example of spatial data set with spatial instance utilities.

Table 1. The utilities of the instances and features of the data set in Fig. 1.

Feature	Instance	Feature	Instance	Feature	Instance	Feature	Instance
	$A1^{15}$		$B1^{9}$		$C1^{17}$		$D1^{16}$
	$A2^{6}$		$B2^{16}$		$C2^{14}$		$D2^{8}$
	$A3^{8}$		$B3^{13}$		$C3^{6}$		$D3^{2}$
	$A4^{2}$		$B4^{5}$		$C4^{6}$		
	$A5^{11}$						
u(A)	42	u(B)	43	u(C)	43	u(D)	26

Definition 3 *(Co-location pattern and co-location instance). A co-location pattern c is a subset of spatial feature set F, i.e.,* $c = \{f_1, ..., f_k\} \subseteq F$. *The number of the features in c is k and it is called the size of c, i.e., c is a size k pattern. A co-location instance* $CI(c)$ *of a co-location pattern c is a set of spatial instances that includes the instances of all feature types in c and these instances have the neighbor relationship R, i.e.,* $CI(c) = \{o_1, ..., o_k\}$. *The set of all co-location instances of c is the table instance of c and denoted as* $TI(c) = \{CI_1(c), ..., CI_p(c)\}$.

Definition 4 *(Utility of a feature in a pattern). The utility of a feature* f_t *in a co-location pattern* $c = \{f_1, ..., f_k\}$, *i.e.,* $f_t \in c$, *is the sum of utilities of the instances belonging to* f_t *in the table instance of c and denoted as*

$$u(f_t, c) = \sum_{o_i^v \in TI(c), f(o_i^v) = f_t} u(o_i^v) \tag{2}$$

Definition 5 *(Intra-utility ratio). The intra-utility ratio of a feature* f_t *in a co-location pattern* $c = \{f_1, ..., f_k\}$, *i.e.,* $f_t \in c$, *is the proportion of the utility of* f_t *in c to its utility in the data set and denoted as*

$$intraUR(f_t, c) = \frac{u(f_t, c)}{u(f_t)} \tag{3}$$

Definition 6 *(Inter-utility ratio). The inter-utility ratio of a feature* f_t *in a co-location pattern* $c = \{f_1, ..., f_k\}$, *i.e.,* $f_t \in c$, *is defined as*

$$interUR(f_t, c) = \frac{\sum_{f_h \in c, f_h \neq f_t} u(f_h, c)}{\sum_{f_h \in c, f_h \neq f_t} u(f_h)} \tag{4}$$

Definition 7 *(Utility participation ratio). The utility participation ratio of a feature* f_t *in a pattern* $c = \{f_1, ..., f_k\}$, *i.e.,* $f_t \in c$, *is the combination of the intra-utility ratio and the inter-utility ratio of the feature and is denoted as*

$$UPR(f_t, c) = \alpha \times intraUR(f_t, c) + \beta \times interUR(f_t, c) \tag{5}$$

where α *and* β *represent the weighted values of the intra-utility ratio and the inter-utility ratio, respectively.*

Definition 8 *(Utility participation index). The utility participation index of a co-location pattern $c = \{f_1, ..., f_k\}$ is the minimum utility participation ratio among all the features in c and denoted as*

$$UPI(c) = \min_{f_t \in c}\{UPR(f_t, c)\} \tag{6}$$

Definition 9 *(High co-location pattern). A co-location pattern $c = \{f_1, ..., f_k\}$ is a high utility co-location pattern if and only if its utility participation index is not smaller than a minimum utility threshold μ given by users, i.e., $UPI(c) \geq \mu$.*

When a co-location pattern has not yet been determined to be a HUCP, the pattern is called candidate co-location pattern. For example, as shown in Fig. 1, examine candidate {A, B, D}, based on Fig. 1, its table instance is {A5^{11}, B3^{13}, D1^{16}} and {A2^{6}, B2^{16}, D2^{8}}, based on equations from (5)–(8), we can compute $UPI(\{\text{A, B, D}\}) = 0.49$. If a user sets $\mu = 0.2$, since $UPI(\{\text{A, B, D}\}) = 0.49 > \mu = 0.2$, thus {A, B, D} is a high utility co-location pattern.

Lemma 1. *UPI does not satisfy the downward closure property.*

Proof. In Fig. 1, when the minimum utility threshold is set to 0.2, i.e., $\mu = 0.2$, we can obtain {A, B, D} is a HUCP since $UPI(\{\text{A, B, D}\})$=0.49. Beside, its a superset {A, B, C, D} and its a subset {A, D} are also HUCPs since $UPI(\{\text{A, B, C, D}\}) = 0.5 > \mu$ and $UPI(\{\text{A, D}\})$=0.74 $> \mu$. It can be found that the UPI value of a pattern can be greater or less than its supersets.

Definition 10 *(ϵ-closed high utility co-location pattern). A co-location pattern $c = \{f_1, ..., f_k\}$ is an ϵ-closed HUCP if and only if c is a HUCP and there exists no HUCP c' such that $c' = \{f_1, ..., f_k\} \cup \{f_{k+1}\}$, i.e., $c \subset c'$, and $|UPI(c) - UPI(c')| \leq \epsilon$, where ϵ is a UPI tolerance value given by users.*

It is easy to find that, when $\epsilon = 0$, ϵ-closed high utility co-location patterns are closed high utility co-location patterns.

Definition 11 *(UPI error rate). Given HUCPS and ϵHUCPS are the set of all HUCPs and the set of ϵ-closed HUCPs, when using ϵ-closed HUCPs to represent HUCPs, the UPI error rate is computed as*

$$\sigma = \sum \frac{|UPI(c') - UPI_{c \in \chi}(c)|}{|HUCPS|} \tag{7}$$

where χ is a set of HUCPs that are removed by ϵ-closed in Definition 10.

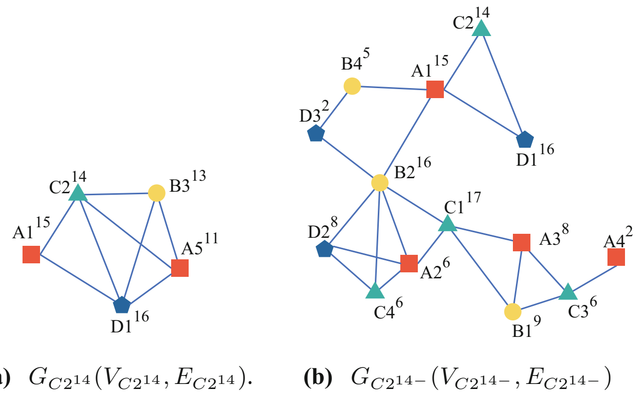
4 Neighboring Instance-Based Mining Algorithm

Based on Lemma 1, since the downward closure property is not held in the UPI metric, if we employ the level-wise search mining framework [19], unnecessary candidates cannot be effectively pruned, thus mining performance is extremely inefficient, especially when the data set is dense. Moreover, the key of discovering ϵ-closed HUCPs is collecting co-location instances of patterns and this step is the most expensive [19]. Thus, reducing candidate search space and quickly collecting co-location instances are the keys of improving mining ϵ-closed HUCPs. This section develops a neighboring instance-based mining ϵ-closed HUCP algorithm.

4.1 Maximal Clique Enumeration

Definition 12 *(Neighboring graph). Given a data set S and a neighbor relationship R, after materializing the neighbor relationship, we get a connected undirected graph $G(V, E)$ with V is the set of vertices of G that are the all instances in S and E is set of edges that are these lines connecting neighboring instances.*

Definition 13 *(Subgraph and remaining neighboring graph). Given an instance o_i and its neighboring instance set $NR(o_i)$, a subgraph of the neighboring graph $G(V, E)$ is $G_{o_i}(V_{o_i}, E_{o_i}) \in G(V, E)$, where $V_{o_i} = \{o_i\} \cup NR(o_i)$ and E_{o_i} is the set of lines that connect neighboring instances in V_{o_i}. The remaining neighboring graph is $G_{o_i^-}(V_{o_i^-}, E_{o_i^-})$ is the graph that contains all vertices and edges in $G(V, E)$ but not in $G_{o_i}(V_{o_i}, E_{o_i})$.*



C2^14
B3^13
A1^15
A5^11
D1^16
C2^14
B4^5
A1^15
D3^2
D1^16
B2^16
C1^17
D2^8
A3^8
A4^2
A2^6
C4^6
B1^9
C3^6
(a) $G_{C2^{14}}(V_{C2^{14}}, E_{C2^{14}})$.
(b) $G_{C2^{14-}}(V_{C2^{14-}}, E_{C2^{14-}})$

Fig. 2. The subgraph and remaining neighboring graph of C2^{14}.

For example, after materializing the spatial neighbor relationship, a graph is obtained as shown in Fig. 1. For C2^{14}, its subgraph and the remaining neighboring graph are plotted in Fig. 2.

After partitioning the neighboring graph by the subgraph and the remaining neighboring graph, a maximal algorithm is applied to the subgraph to enumerate maximal cliques (MCs) [5,6]. Here we do not focus on designing a new MC enumeration algorithm, this work is paid attention to how to employ MCs to improve the performance of mining ϵ-closed HUCPs. Algorithm 1 describes the pseudocode of a MC enumeration algorithm that is developed by Eppstein et al. [6]. This algorithm inputs a set of vertices and edges, then uses a recursive process to find maximal cliques. For more details, please reference [6].

Combining the partitioning strategy and MC enumeration algorithm, Algorithm 2 shows the pseudocode of listing all MCs from the input spatial data set under a neighbor relationship R. This algorithm first materializes the neighbor

Algorithm 1. Enumerating maximal cliques on a graph

```
    Input: $G(V, E)$
    Output: maximal cliques, $MCs$
    $P \leftarrow V, R \leftarrow \emptyset, X \leftarrow \emptyset$
    Proc: BronKerboschDegeneracy($V, E$)
 1: for each vertex $v_i$ in a degeneracy ordering of $G(V, E)$ do
 2:     $P \leftarrow NR(v_i) \cap \{v_{i+1}, ..., v_{n-1}\}$
 3:     $X \leftarrow NR(v_i) \cap \{v_0, ..., v_{i-1}\}$
 4:     BronKerboschPivot($P, \{v_i\}, X$)
 5: end for
    Proc: BronKerboschPivot($P, R, X$)
 1: if $P \cup X = \emptyset$ then
 2:     $MCs = MCs \cup R$                              ▷ $R$ is a maximal clique
 3: end if
 4: Choose a pivot $u \in P \cup X$ to maximize $|P \cup NR(u)|$
 5: for $v \in P \setminus NR(u)$ do
 6:     BronKerboschPivot($P \cap NR(v), R \cup \{v\}, X \cap NR(v)$)
 7:     $P \leftarrow P \setminus v$
 8:     $X \leftarrow X \cup v$
 9: end for
10: return $MCs$
```

Algorithm 2. Enumerating MCs

```
    Input: $G(V, E)$
    Output: a set of maximal cliques, $MCS$
 1: $NRS = \{..., NR(o_i), ...\} \leftarrow$ materializing_neighbor_relationship($S, R$)
 2: while $S \neq \emptyset$ do
 3:     $o_i \leftarrow S$.pop()
 4:     $G_{o_i}(V_{o_i}, E_{o_i}) \leftarrow$generate_subgraph($o_i, NRS$)
 5:     $G_{o_i^-}(V_{o_i^-}, E_{o_i^-}) \leftarrow$generate_ remaining_neighboring_graph($G_{o_i}(V_{o_i}, E_{o_i}), S$)
 6:     $S \leftarrow V_{o_i^-}$
 7:     $MCs \leftarrow$ Algorithm 1
 8:     $MCS \leftarrow MCS \cup MCs$
 9: end while
10: return $MCS$
```

relationship between instances in S (Step 1). Then, for each instance $o_i \in S$, it constructs the subgraph $G_{o_i}(V_{o_i}, E_{o_i})$ (Step 4) and the remaining neighboring graph $G_{o_i^-}(V_{o_i^-}), E_{o_i^-})$ (Step 5). After that, Algorithm 1 is employed to find MCs in $G_{o_i}(V_{o_i}, E_{o_i})$ (Step 6). Finally, Algorithm 2 returns a set of MCs (Step 9).

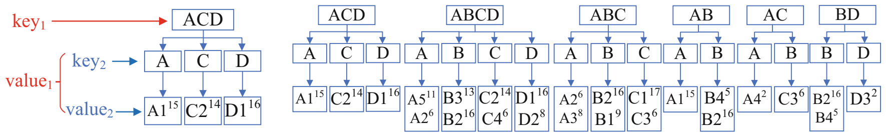
Table 2 lists all MCs enumerated from the neighboring graph in the Fig. 1.

4.2 Co-location Instance Hash Table

Based on Definition 8, when calculating the utility participation ratio of a pattern $c = \{f_1, ...f_k\}$, it only needs to know the spatial instances of each feature

Table 2. The MCs listed from Fig. 1.

MC	MC	MC
{A1, C2, D1}	{B2, D3}	{A3, B1, C1}
{A5, B3, C2, D1}	{B4, D3}	{A3, B1, C3}
{A1, B4}	{A2, B2, C4, D2}	{A4, C3}
{A1, B2}	{A2, B2, C1}	



(a) An item. **(b)** All items.

Fig. 3. The co-location instance hash table constructed based on MCs of the data set in Fig. 1.

participating in this pattern, it does not care which instance has a neighbor relationship with which instances. Therefore, given a pattern, if we can query the participation instances of each feature into the pattern, the utility participation ratio of the pattern can be easily calculated. We arrange these maximal cliques into a special hash table structure.

Definition 14 *(Co-location instance hash table). A co-location instance hash table is a two-level hash table, i.e., $(key_1, (key_2, value_2))$, where key_1 is the set of spatial feature types of these instances in a MC, key_2 is the spatial feature type of each instance in MC, and $value_2$ is the instance itself in MC. All MCs that instances in them belong to the same spatial feature type set are grouped into an item in the co-location hash table.*

Figure 3(a) shows an example of an item in the co-location instance hash table that is constructed from maximal clique $A1^{15}$, $C2^{14}$, $D1^{16}$. Based on Table 2, the co-location instance hash table of the input data set under the neighbor relationship is plotted in Fig. 3(b).

4.3 Mining ϵ-Closed HUCPs

After constructing the co-location instance hash table based on MCs of the input data set, the next step is how to quickly collect the instances that participate in the co-location instances of each pattern from the hash table structure.

Lemma 2. *Given a co-location pattern $c = \{f_1, ...f_k\}$, the instances of each feature type in c that participate in the table instance of c are collected from the values in the co-location instance hash table corresponding to the keys that are supersets of c (including if c is a key, the key is also a superset of c).*

Algorithm 3. Filtering ϵ-closed HUCPs

```
Input: the co-location instance hash table, $CIHash$; $d, \mu, \epsilon$
Output: the set of $\epsilon$-closed HUPC, $\epsilon HUCPS$
1: $keyset \leftarrow CIHash$.getKeys()
2: while $keyset \neq \emptyset$ do
3:     $keyset \leftarrow$sort_by_size($keyset$)
4:     $c \leftarrow keyset$.pop()
5:     for $item \in CIHash$ do
6:         if $c \subseteq item$.getKey() then
7:             $TI(c) \leftarrow$get_values($item$.getValue)
8:         end if
9:     end for
10:    $UPI(c) \leftarrow$calculate_UPI($TI(c)$)
11:    if $UPI(c) \geq \mu$ then
12:        for $c' \in \epsilon HUCPS$ do
13:            if $(c \subseteq c')$ and $|UPI(c) - UPI(c')| > \epsilon$ then
14:                $\epsilon HUCPS$.add($c$)
15:                break
16:            end if
17:        end for
18:    end if
19:    $subc \leftarrow$generate_direct_subsets($c$)
20:    $keyset \leftarrow keyset \cup subc$
21: end while
22: return $\epsilon HUCPS$
```

Proof. From the definition of the co-location instance hash table, $value_2$ is a set of participating instances belonging to the feature type be key_2 in a co-location pattern named key_1. If a pattern $c \equiv key_1$, we can get the participating instances of c directly from $value_2$. However, the co-location instances of c may be the subsets of MCs that are built into an item in the co-location hash table, thus a part of co-location instances of c is queried from the values corresponding to the keys that are supersets of c.

For example, for co-location pattern $c = \{$A, B$\}$, the instances that participate in c of A and B are obtained from the values of keys AB, ABC, and ABCD, i.e., $\{A1^{15}, A2^{6}, A3^{8}, A5^{11}\}$ and $\{B2^{16}, B3^{13}, B4^{5}\}$, respectively.

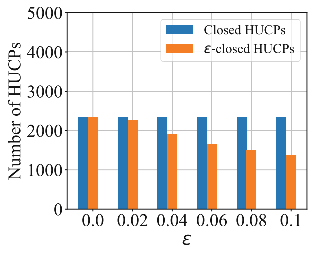
Algorithm 3 describes the pseudocode of filtering ϵ-closed HUCPs from the co-location instance hash table. This algorithm first gets all keys in the co-location instance hash table and stores them in a set of keys, *keyset* (Step 1). Next, Algorithm 3 executes a while loop. In the loop, *keyset* is sorted in descending order size of the keys (Step 3) and it gets the first key as a co-location pattern c (Step 4). Then, it queries these keys that are a superset of c (Steps 5–6), and then takes the corresponding values and puts them into the table instance of c (Step 7). After that, the utility participation index of c is calculated (Step 8). If c is a HUCP (Step 9), it needs to verify whether the pattern is ϵ-closed (Steps 10–13). Next, the direct subsets of c are generated and put into *keyset* as new candidates that need to be examined (Steps 14–15). And finally, a set of ϵ-closed HUCPs is returned to users (Step 16).

5 Experiments

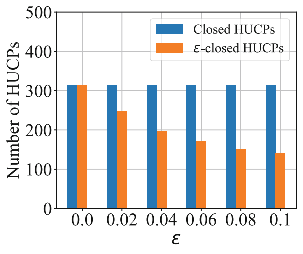
A set of experiments is constructed in this section to evaluate the effectiveness and efficiency of the proposed method. Our work is compared with the original work about mining HUCPs [19]. The two algorithms are implemented in C++ and run on a computer with CPU Intel(R) Core i7-3770 and 16 GB of RAM.

Table 3. A summary of the experimental data sets.

Name	Spatial frame size	# instances	# features	Property
Shenzhen [6]	28,800 × 89,900 (m^2)	35,220	20	Dense, clustering
Shanghai [6]	65,500 × 113,600 (m^2)	41,450	15	Dense, uniform
Vegetation [18]	71,930 × 66,440 (m^2)	503,340	16	Dense
Synthetic [22]	5,000 × 5,000	*	15	Dense



(a) Shenzhen (d=250m,$\mu = 0.1$).



(b) Shanghai (d=350m,$\mu = 0.1$).

Fig. 4. The number of HUCPs with the increase of the tolerance ϵ.

5.1 Spatial Data Sets

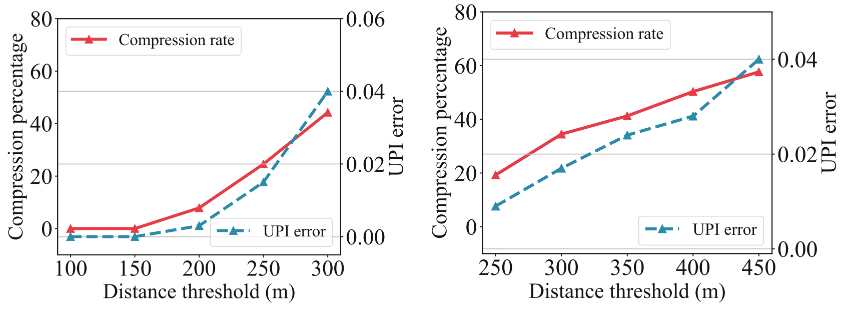
Both synthetic and real data sets are used in our experiments, in which the synthetic data sets are generated by a data generator [22] and the real data sets are listed in Table 3. As can be seen from Table 3, the experimental data sets are dense and have different numbers of instances, numbers of features, and distributions. This enables a more complete investigation of the performance of the proposed algorithm. The utility of the instances in these data sets is generated according to an exponential distribution under a hypothesis that the number of the low utility instances is much more than the high utility instances.

5.2 The Number HUCPs

The first experiment examines the number of HUCPs under different values of the tolerance ϵ. Since closed HUCPs are a lossless compression of all HUCPs, we only need to compare the proposed compression method with the closed. As shown in Fig. 4, many patterns can be reduced effectively by giving a relatively small tolerance value, ϵ, and as the value of the tolerance ϵ increases, more and more patterns are removed. Because the condition of the closed is too strict, and according to the UPI definition, this condition is difficult to achieve. It can be seen that reducing HUCPs can be effectively achieved by relaxing the closure condition under the tolerance ϵ controlled by users.

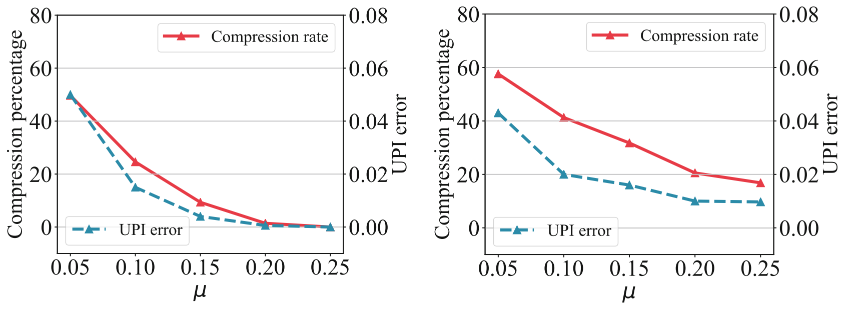
5.3 Compression Rate and UPI Error Rate

In this experiment, the compression rate and the UPI error rate of ϵ-closed HUCPs are surveyed to demonstrate the power of the ϵ-closed. The compression rate is calculated based on the closed HUCPs. We fix $\epsilon = 0.05$ (a sufficiently low tolerance in our opinion). Figure 5 plots the results on the two real data sets under different distance thresholds. As the distance threshold increases, the compression ratio increases, and the UPI error rate also increases. But the UPI error rate does not exceed the configured tolerance. The above results prove that the ϵ-closed can balance the compression rate and the UPI error rate, and it can give a large pattern compression rate within a relatively small range of the UPI error rate.



(a) Shenzhen ($\epsilon = 0.05, \mu = 0.1$). **(b)** Shanghai ($\epsilon = 0.05, \mu = 0.1$).

Fig. 5. The compression rate and the error rate of the ϵ-closed method under different distance thresholds.



(a) Shenzhen ($\epsilon = 0.05, d = 250$m). **(b)** Shanghai ($\epsilon = 0.05, d = 350$m).

Fig. 6. The compression rate and the error rate of the ϵ-closed method under different minimum utility thresholds.

Moreover, the compression rate and the UPI error rate under different minimum utility thresholds are also examined and plotted in Fig. 6. As shown in Fig. 6, in the small value of minimum utility thresholds, the compression rate is large and the UPI error rate is also large. The larger the minimum utility threshold, the greater the difference in the UPI values of the patterns, so the number of HUCPs that can be reduced will be less.

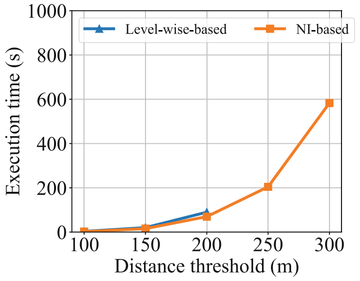
From the above experimental results, we can conclude that the denser the HUCPs (large values of distance thresholds and small values of minimum utility thresholds), the stronger the ϵ-closed compression ability.

5.4 Scalability

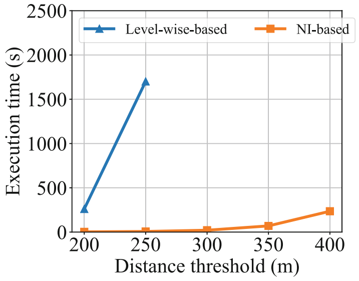
In this section, the scalability of the proposed neighboring instance-based mining algorithm is evaluated with several workloads, i.e., different distance thresholds, minimum high utility thresholds, and numbers of spatial instances. For the convenience of description, the mining HUCP algorithm proposed by Wang et al. [19] (to the best of our knowledge, this is the only work on mining HUCP so far) is named level-wise-based since it adopts the traditional mining framework [22]. And our neighboring instance-based algorithm is named NI-based for short.

Figure 7 describes the execution time of the comparing algorithms on different distance thresholds. As can be seen that at a small distance threshold, the efficiency of the comparing algorithms is the same, the level-wise-based algorithm fails in the case of large distance thresholds, while the neighboring instance-based mining algorithm shows good scalability.

The execution of the proposed algorithm under different minimum high utility thresholds is also plotted in Fig. 8. As can be seen that our algorithm shows better performance at small thresholds.

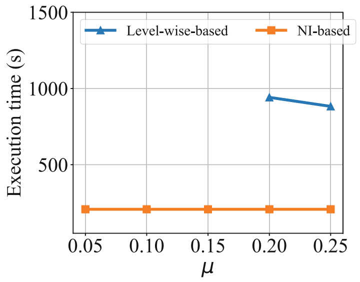


(a) Shenzhen ($\epsilon = 0.05, \mu = 0.1$).

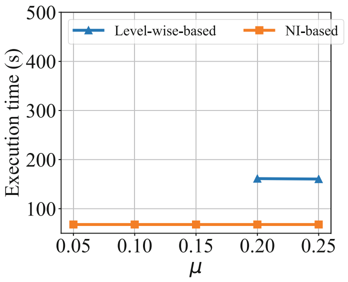


(b) Shanghai ($\epsilon = 0.05, \mu = 0.1$).

Fig. 7. The execution time of the comparing algorithms on different distance thresholds.



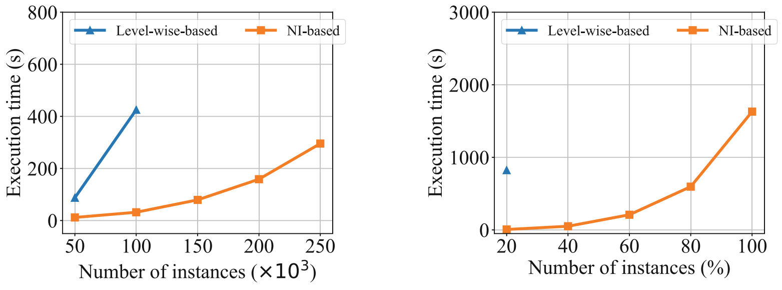
(a) Shenzhen ($\epsilon = 0.05, d = 250$m).



(b) Shanghai ($\epsilon = 0.05, d = 350$m).

Fig. 8. The execution time of the comparing algorithms on different minimum utility thresholds.

Moreover, Fig. 9 plots the execution of the two algorithms on different numbers of spatial instances. To generate different sizes of input data sets, for the synthetic data set, we fix the number of features and change the number of instances, whereas we perform random sampling on the real data set (Vegetation data set). As shown in Fig. 9, the execution time of the level-wise-based algorithm increases dramatically with the increase of the number of spatial instances. While the proposed mining algorithm gives scalability to large dense data sets.



(a) Synthetic ($\epsilon = 0.05, d = 30$). **(b)** Vegetation ($\epsilon = 0.05, d = 800$m).

Fig. 9. The execution time of the comparing algorithms on different numbers of spatial instances ($\mu = 0.1$ for all).

6 Conclusion

This work proposes a concise representation of HUCPs, ϵ-closed HUCPs. This representation allows a pattern and its superset to have a small tolerance ϵ that users can control according to their applications. Experimental results on the real data sets show that this representation can give a good compression rate under a small loss of information. Moreover, to avoid too many unnecessary candidates that need to be examined due to the UPI metric not satisfying the downward closure property, this work proposes a neighboring instance-based mining algorithm. The algorithm performs better than the existing algorithm on the experimental data sets.

The proposed mining algorithm can perform the parallel computation to improve the mining efficiency in giant spatial data sets.

Acknowledgements. This work is supported by the National Natural Science Foundation of China (61966036), the Project of Innovative Research Team of Yunnan Province (2018HC019), the Yunnan Fundamental Research Projects (202201AS070015).

References

1. Akbari, M., Samadzadegan, F., Weibel, R.: A generic regional spatio-temporal co-occurrence pattern mining model: a case study for air pollution. Journal of Geographical Systems **17**(3), 249–274 (2015)
2. Andrzejewski, W., Boinski, P.: Parallel approach to incremental co-location pattern mining. Inf. Sci. **496**, 485–505 (2019)
3. Bao, X., Wang, L.: A clique-based approach for co-location pattern mining. Inf. Sci. **490**, 244–264 (2019)
4. Cai, J., Liu, Q., Deng, M., Tang, J., He, Z.: Adaptive detection of statistically significant regional spatial co-location patterns. Comput. Environ. Urban Syst. **68**, 53–63 (2018)

5. Chang, L.: Efficient maximum clique computation over large sparse graphs. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 529–538 (2019)
6. Eppstein, D., Löffler, M., Strash, D.: Listing all maximal cliques in large sparse real-world graphs in near-optimal time. J. Exp. Algorithmics (JEA) **18**, 1–3 (2013)
7. He, Z., Deng, M., Xie, Z., Wu, L., Chen, Z., Pei, T.: Discovering the joint influence of urban facilities on crime occurrence using spatial co-location pattern mining. Cities **99**, 102612 (2020)
8. Huang, Y., Shekhar, S., Xiong, H.: Discovering colocation patterns from spatial data sets: a general approach. IEEE Trans. Knowl. Data Eng. **16**(12), 1472–1485 (2004)
9. Lee, I., Phillips, P.: Urban crime analysis through areal categorized multivariate associations mining. Appl. Artif. Intell. **22**(5), 483–499 (2008)
10. Leibovici, D.G., Claramunt, C., Le Guyader, D., Brosset, D.: Local and global spatio-temporal entropy indices based on distance-ratios and co-occurrences distributions. Int. J. Geogr. Inf. Sci. **28**(5), 1061–1084 (2014)
11. 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**(4), 651–692 (2016)
12. Liu, Q., Liu, W., Deng, M., Cai, J., Liu, Y.: An adaptive detection of multilevel co-location patterns based on natural neighborhoods. Int. J. Geogr. Inf. Sci. **35**(3), 556–581 (2021)
13. Luna, J.M., Fournier-Viger, P., Ventura, S.: Frequent itemset mining: a 25 years review. Wires Data. Min. Knowl. **9**(6), e1329 (2019)
14. Raj, S., Ramesh, D., Sreenu, M., Sethi, K.K.: Eafim: efficient apriori-based frequent itemset mining algorithm on spark for big transactional data. Knowl. Inf. Syst. **62**(9), 3565–3583 (2020)
15. Sheshikala, M., Rao, D.R., Prakash, R.V.: A map-reduce framework for finding clusters of colocation patterns-a summary of results. In: 2017 IEEE 7th International Advance Computing Conference (IACC), pp. 129–131. IEEE (2017)
16. Tran, V., Wang, L., Chen, H., Xiao, Q.: MCHT: a maximal clique and hash table-based maximal prevalent co-location pattern mining algorithm. Expert Syst. Appl. **175**, 114830 (2021)
17. Tran, V., Wang, L., Zhou, L.: A spatial co-location pattern mining framework insensitive to prevalence thresholds based on overlapping cliques. Distributed Parallel Databases 1–38 (2021)
18. Wang, L., Bao, X., Chen, H., Cao, L.: Effective lossless condensed representation and discovery of spatial co-location patterns. Inform. Sci. **436**, 197–213 (2018)
19. Wang, L., Jiang, W., Chen, H., Fang, Y.: Efficiently mining high utility co-location patterns from spatial data sets with instance-specific utilities. In: Candan, S., Chen, L., Pedersen, T.B., Chang, L., Hua, W. (eds.) DASFAA 2017. LNCS, vol. 10178, pp. 458–474. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-55699-4_28
20. 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. Inf. Sci. **542**, 357–379 (2021)
21. Yoo, J.S., Bow, M.: A framework for generating condensed co-location sets from spatial databases. Intell. Data Anal. **23**(2), 333–355 (2019)
22. Yoo, J.S., Shekhar, S.: A joinless approach for mining spatial colocation patterns. IEEE Trans. Knowl. Data Eng. **18**(10), 1323–1337 (2006)