



A Parallel Spatial Co-location Pattern Mining Approach Based on Ordered Clique Growth

Peizhong Yang, Lizhen Wang^(✉), and Xiaoxuan Wang

School of Information Science and Engineering, Yunnan University,
Kunming 650091, China
pzyang0924@163.com, lzhwang@ynu.edu.cn

Abstract. Co-location patterns or subsets of spatial features, whose instances are frequently located together, are particularly valuable for discovering spatial dependencies. Although lots of spatial co-location pattern mining approaches have been proposed, the computational cost is still expensive. In this paper, we propose an iterative mining framework based on MapReduce to mine co-location patterns efficiently from massive spatial data. Our approach searches for co-location patterns in parallel through expanding ordered cliques and there is no candidate set generated. A large number of experimental results on synthetic and real-world datasets show that the proposed method is efficient and scalable for massive spatial data, and is faster than other parallel methods.

Keywords: Spatial data mining · Co-location patterns · Ordered clique
Parallel algorithm · MapReduce

1 Introduction

The spatial co-location pattern mining is one of the spatial knowledge discovery technologies, and it is intended to discover a subset of spatial features whose instances are frequently located together. The spatial co-location pattern mining has many applications [1], and various co-location pattern mining methods have been proposed. But most methods are serial processing and inefficient when handling massive spatial data. As a solution, the methods of parallel co-location pattern mining are imperative. However, little research pays attention to the parallel co-location pattern mining. In this work, we propose a parallel spatial co-location pattern mining approach based on ordered clique growth, and it needs not to generate candidate sets and check clique instances. The algorithm is implemented on Apache Spark, and extensive experiments are conducted to evaluate the efficiency. Experimental results demonstrate that our method is efficient and scalable for mining co-location patterns from massive spatial data.

The main contributions of this work can be summarized as follows: (1) The ordered clique expanding method in a level-wise manner is proposed. (2) An iterative framework for spatial co-location pattern mining based on ordered clique growth is provided. (3) We suggest a pruning strategy to cut out some ordered cliques early to speed up the mining process. (4) A parallel spatial co-location pattern mining algorithm based on MapReduce is proposed.

The rest of this paper is organized as follows: Sect. 2 reviews related work. Section 3 presents the basic concept of spatial co-location mining and the MapReduce. In Sect. 4, a novel parallel spatial co-location pattern mining method is provided. Section 5 shows experimental evaluations and the paper will conclude on Sect. 6.

2 Related Work

A large number of methods have been developed to discover co-location patterns. Huang et al. [1] defined the participation index to measure the prevalence of co-location patterns and proposed the *Apriori-like* method join-based. Based on the participation index, a partial-join approach [3] and a join-less approach [2] proposed by Yoo and Shekhar to improve mining efficiency. However, above methods are difficult to avoid generating huge candidate patterns and storing massive clique instances. To address this problem, various optimization algorithms have been proposed [4–6, 8], but these methods are serial processing and they are inefficient when processing massive spatial data. Yoo et al. [7] proposed a parallel spatial co-location pattern mining algorithm based on MapReduce. It can handle massive spatial data, but the candidate clique instances generation and the clique checking operation are still time-consuming. In this work, a novel parallel spatial co-location pattern mining approach based on ordered clique growth is proposed. There is no candidate set generated and the clique testing operations can be avoided in our method.

3 Basic Concepts

In a spatial database, let $F = \{f_1, f_2, \dots, f_m\}$ be a set of spatial features and $O = \{o_1, o_2, \dots, o_n\}$ be a set of instances of F . Two instances have the spatial neighbor relationship R , if the distance between two instances is less than a threshold d . A co-location pattern $Cl = \{f_1 \dots f_k\}$ is a subset of spatial features, whose instances frequently form clique under R . A set of instances I is a row instance of Cl , if (1) I contains all features of Cl and no proper subset of I does so, and (2) all instances of I form a clique. The set of all row instances of Cl is called table instance, denoted as $T(Cl)$. The *Participation Index* (PI) is defined [1] to evaluate the prevalence of co-location pattern. The participation index of Cl is the minimum *Participation Ratio* (PR) for all spatial features in Cl . The participation ratio of f_i in Cl can be computed as:

$$PR(Cl, f_i) = \frac{\text{number of distinct instances of } f_i \text{ in } T(Cl)}{\text{number of global instances of } f_i} \quad (1)$$

Cl is called a prevalent co-location pattern, if the participation index of Cl is not less than a given prevalence threshold min_prev . The participation index is anti-monotone [1], which means the PI of a pattern is not bigger than the PI of its sub-patterns.

MapReduce is a programming model which provides a highly scalable and flexible framework for data-oriented parallel computing. A MapReduce job is executed in two main phases of user defined data transformation functions, *map* and *reduce*. In the first

phase, the *key-value* pairs are processed by Mapper instances and the output of the *map* function is another set of intermediate *key-value* pairs. The intermediate process of moving the intermediate *key-value* pairs from the map tasks to the assigned Reducer is called *shuffle* phase. At the completion of the *shuffle*, all *values* associated with the same *key* are fed to a Reducer and processed by the reduce function. Each Reducer generates a set of new *key-value* pairs as the output of the job.

4 A Parallel Approach

The generation of table instance set is the most time-consuming operation in co-location pattern mining. Generating row instance in parallel is an effective way to improve the efficiency of the mining process. This section presents a method to discover co-location patterns through expanding ordered clique in parallel.

4.1 Ordered Clique Growth

Definition 1 (Ordered Clique). Given an instance set $I_k = \{o_1, o_2, \dots, o_k\}$, $o_i \in O$, $1 \leq i \leq k$, if $f(o_i) \leq f(o_j)$ and $R(o_i, o_j)$ holds for every $1 \leq i \leq j \leq k$, I_k is called an ordered clique. The set I_k contains k instances, thus I_k is a size- k ordered clique.

$f(o_i)$ is the feature type of instance o_i . $f(o_i) \leq f(o_j)$ represents that the feature type of o_i is not greater than o_j in alphabetical order.

Definition 2 (Neighbor Set of Instance). Given an instance $o_i \in O$, the neighbor set of instance o_i is defined as:

$$NSI(o_i) = \{o_k | o_k \in O \wedge R(o_i, o_k) \wedge (f(o_i) < f(o_k))\}$$

That is, the neighbor set of o_i consists of some instances who has spatial neighbor relationship with o_i and whose feature type is bigger than o_i in alphabetical order.

Definition 3 (Neighbor Set of Clique). Given a size- k ordered clique $I_k = \{o_1, o_2, \dots, o_k\}$, $o_i \in O$, $1 \leq i \leq k$. The neighbor set of clique I_k is defined as:

$$NSC(I_k) = NSI(o_1) \cap \dots \cap NSI(o_k)$$

Lemma 1. Given a size- k ordered clique I_k , if $o \in NSC(I_k)$, appending o to I_k constitutes a new clique $I_{k+1} = \{o_1, o_2, \dots, o_k, o\}$, and I_{k+1} is a size- $(k + 1)$ ordered clique.

Proof. If $o_i \in I_k$, $1 \leq i \leq k$, and $o \in NSC(I_k)$. Thus, $o \in NSI(o_i)$ by Definition 3, that is, the instance o has spatial neighbor relationship with all instances in I_k , and the feature type of o is bigger than all instances in I_k lexicographically according to Definition 2. Therefore, appending o to I_k to form a size- $(k + 1)$ clique I_{k+1} , and it is ordered clique by definition.

4.2 An Iterative Parallel Mining Framework

According to Lemma 1, given an ordered clique I_k , the size- $(k + 1)$ ($k \geq 1$) ordered cliques prefixed with I_k can be produced by appending an element in $NSC(I_k)$ to I_k . Obviously, the operation of expanding ordered cliques can be performed in parallel. A size- k ordered clique corresponds to a row instance of a size- k ordered co-location pattern. When all size- k ordered cliques are generated, table instances for all size- k ordered co-location patterns can be collected easily. Starting from the size-2 ordered cliques, we can search for all ordered co-location patterns level by level. Naturally, an iterative parallel co-location pattern mining framework based on ordered clique growth is suggested. The mining framework is given in Fig. 1. In addition, a pruning strategy is proposed to narrow searching space.

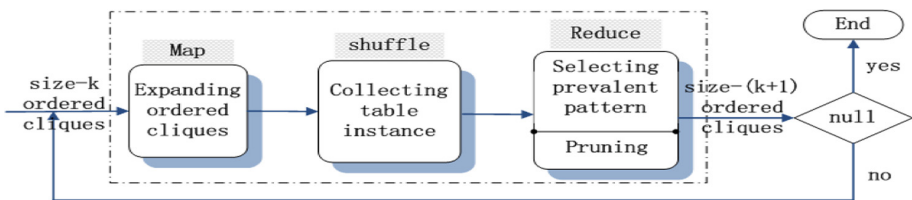


Fig. 1. The mining framework

Pruning Strategy. Given a size- k ordered clique I_k , and I_k is a row instance of the pattern Cl . If Cl is not prevalent, the size- $(k + 1)$ ordered cliques prefixed with I_k can be pruned.

Depending on the anti-monotone property of the participation index, if the pattern Cl is not prevalent, the super set of Cl must be not prevalent. A size- $(k + 1)$ ordered clique prefixed with I_k corresponds to a size- $(k + 1)$ co-location pattern Cl' , and Cl' must be the super set of Cl . We do not need to search for the pattern Cl' because the pattern Cl is not prevalent, thus the size- $(k + 1)$ ordered cliques who are prefixed with I_k can be pruned.

4.3 Parallel Algorithm

In this subsection, the parallel co-location pattern mining algorithm based on ordered clique growth (PCPM_OC) is presented. The algorithm is described by MapReduce procedure presented in Fig. 2.

Firstly, we count and store the number of instances per spatial feature utilizing Job1 presented in Fig. 2(a). It is preparation for future participation index calculation.

The task of generating the neighbor set of instance is accomplished by Job2 presented in Fig. 2(b). The pair of instances who have spatial neighbor relationship is the input for Mapper. Spatial neighbor relationships can be obtained in advance by the parallel neighbor searching method proposed in [7]. Then, the pair $\langle o_i, o_j \rangle$ is the output of Mapper and the feature type of o_i is smaller than o_j lexicographically. At *shuffle* phase, the neighbor of per instance will be collected in a set. The pair, $\langle o, NSI(o) \rangle$, be

fed to Reducer and then the pair will be stored for the operation of expanding ordered clique. In order to construct the initial input of Job3, an instance is considered as a size-1 ordered clique. An ordered clique I consists of 3 parts, the pattern corresponding to I , the instance set of I and the neighbor set of clique. The new *key-value* pair on behalf of a size-1 ordered clique is emitted as the output of Reducer.

The procedure of searching for prevalent co-location patterns is shown in Fig. 2(c). The Mapper is fed with size- k ordered cliques, and performs the operation of expanding ordered clique according to Lemma 1. For a size- k ordered clique I_k , the size- $(k + 1)$ ordered cliques prefixed with I_k are emitted. At the completion of the shuffle, the ordered cliques associated with the same pattern will be gathered. In Reducer, computing the participation index of the pattern, and then storing prevalent pattern and pruning the searching space for the next iteration. Depending on the pruning strategy, just emitting the size- $(k + 1)$ ordered cliques corresponding to prevalent patterns and whose neighbor set is not null. Proceeding from size-1 ordered cliques, all prevalent co-location patterns can be obtained level-by-level through executing Job3 iteratively.

<pre> 1: procedure Mapper(<i>null</i>, <i>value</i>=<i>o</i>) 2: emit($f(o)$, 1) 3: end procedure 4: procedure Reducer (<i>key</i>=feature type , <i>value</i>=[1]) 5: count=sum(<i>value</i>) 6: store(feature, count) 7: emit(feature, count) 8: end procedure </pre>	<pre> 1: procedure Mapper(<i>key</i>=o_n, <i>value</i>=o_i) 2: if $f(o_i) < f(o_n)$ 3: emit(o_n, o_i) 4: else 5: emit(o_n, o_i) 6: end procedure 7: procedure Reducer(<i>key</i>=o, <i>value</i>=$NSI(o)$) 8: store(o, $NSI(o)$) 9: <i>pattern</i>=$f(o)$ $I = o.id$ $NSC(I) = NSI(o)$ 10: emit(<i>pattern</i>, (I, $NSC(I)$)) 11: end procedure </pre>
(a)	(b)
<pre> 1: procedure Mapper(<i>key</i>=<i>pattern</i>, <i>value</i>=(I, $NSC(I)$)) 2: for all $o \in NSC(I)$ do 3: <i>newPattern</i>= <i>pattern</i>+ $f(o)$ 4: <i>newI</i>=$I+o.id$ 5: load neighbor set of o to $NSI(o)$ 6: $NSC(newI) = NSC(I) \cap NSI(o)$ 7: emit(<i>newPattern</i>, (<i>newI</i>, $NSC(newI)$)) 8: end for 9: end procedure 10: procedure Reducer (<i>key</i>=<i>pattern</i>, <i>value</i>=(I, $NSC(I)$)) </pre>	<pre> 11: compute <i>PI</i> of <i>pattern</i> 12: if $PI \geq min_prev$ 13: store(<i>pattern</i>, <i>PI</i>) 14: for all $v \in value$ do 15: if $v.NSC(I) \neq \emptyset$ 16: emit(<i>pattern</i>, ($v.I$, $v.NSC(I)$)) 17: end if 18: end for 19: end if 20: end procedure </pre>
(c)	

Fig. 2. The parallel spatial co-location pattern mining approach based on ordered clique growth, (a) Job1: procedure for counting the number of instances per spatial feature, (b) Job2: procedure for generating the neighbor set of instance, (c) Job3: procedure of searching for prevalent co-location patterns

Our algorithm has a few advantages: (1) High degree of parallelism. The expanding operation for each size- k ordered clique can be carried out independently, and it can resolve the problem that generating table instances is time-consuming in co-location pattern mining. (2) Iterative execution. Because of the iterative mining framework, we

can re-use the previously processed information. (3) Effective pruning. A pruning strategy is proposed to narrow the searching space. (4) No candidate sets. There is not any candidate pattern and candidate clique instance to be generated. (5) No clique checking. Our method ensures that each expanding operation generates ordered cliques.

5 Experimental Evaluation

We implement our algorithm in Spark library functions. The performance evaluation is conducted on the cluster that deployed Hadoop and Spark. The cluster consists of one master node and six worker nodes with the following characteristics: (1) CPU per node: Intel Core i7-6700, @3.40 GHz; (2) Memory per node: 8 GB. Experiments are conducted on real and synthetic data sets. Two real data sets are used. The first is a plant dataset of the “Three Parallel Rivers of Yunnan Protected Areas” that contains 25 spatial features and 13,348 spatial instances. The second is the POI of Beijing that contains 63 spatial features and 303,895 spatial instances. The synthetic data sets are generated based on the spatial data generator described in [1].

5.1 Compared with Serial Mining Methods

In the first experiment, we evaluate the efficiency of the PCPM_OC compared with two state-of-the-art serial co-location mining algorithms, the JoinBase [2] and the JoinLess [3]. We use a small data set, the plant dataset, because the serial algorithms easily overflow on large data sets. In Fig. 3(a), we set the participation index threshold min_prev to 0.1. The running time of the JoinBase increases dramatically with the increasing of the distance threshold d and the efficiency of the JoinLess is slightly better than the PCPM_OC when d is smaller, owing to the PCPM_OC requires extra cost. When $d > 3,000$ m, the advantage of the PCPM_OC can be reflected. In Fig. 3(b), we set d to 3,000 m. The running time of three algorithms decreases with the increasing of min_prev , because fewer patterns need to be searched when min_prev is larger. Similarly, the JoinBase is the least efficient. When min_prev is smaller, more patterns are searched, the PCPM_OC is better than two serial algorithms, because the large number of calculation is executed in parallel.

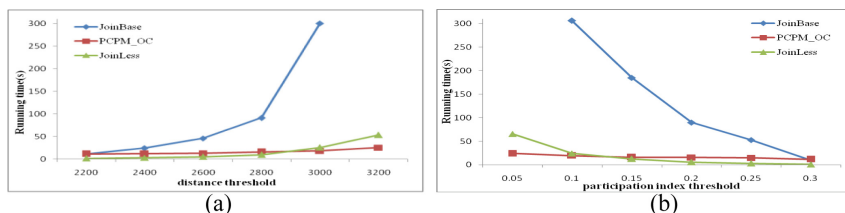


Fig. 3. Running time over the plant dataset: (a) by the distance thresholds, (b) by the participation index thresholds

5.2 Compared with PCPM_SN

In the second experiment, we compare the efficiency of the PCPM_OC with the parallel algorithm (PCPM_SN) proposed in [7]. We implement the PCPM_SN algorithm in Spark and use the data set of the POI of Beijing. First, the effect of the distance threshold is evaluated. In Fig. 4(a), we set min_prev to 0.4. The running time of two methods is increased with the increasing of d , because a larger value of d means more instances could form cliques. The performance of the PCPM_OC is better than the PCPM_SN, as there is no candidate clique generation and clique checking operation in the PCPM_OC. Next, we assess the effect of the participation index thresholds. In Fig. 4(b), we set d to 300 m. As min_prev becomes higher, more patterns dissatisfy the condition of prevalent co-location patterns. Naturally, the performance of two algorithms improves. When min_prev is smaller, the running time of the PCPM_OC is less than the PCPM_SN obviously. Because more patterns are searched means that more candidate clique instances need to be generated and more clique checking operations are required to be undertaken in the PCPM_SN, but these are not required in the PCPM_OC.

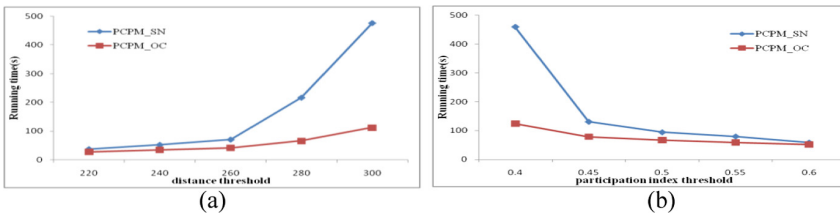


Fig. 4. Running time over the POI of Beijing: (a) by the distance thresholds, (b) by the participation index thresholds.

5.3 Scalability Evaluation

In order to assess the scalability, some experiments are conducted on synthetic data sets. We generate spatial instances and randomly distribute them into a $10,000 \times 10,000$ space, and $min_prev = 0.1$, $d = 20$.

In Fig. 5(a), we assess the effect of the number of spatial features. The average of instances per spatial feature is 10,000. The running time is increased with the increasing of the number of spatial features, because more potential patterns are required to be searched when increasing the number of spatial features. The performance of the PCPM_OC is preferable to the PCPM_SN, especially when the number of spatial features is larger.

In Fig. 5(b), we assess the impact of the number of spatial instances and set the number of spatial features to 100. As the number of spatial instances becomes larger, the running time of two algorithms is raised. The distribution of spatial instances is more densely and more spatial instances could form cliques when the number of total spatial instances is larger. The performance of the PCPM_OC outperforms the PCPM_SN, especially when the number of spatial instances is larger.

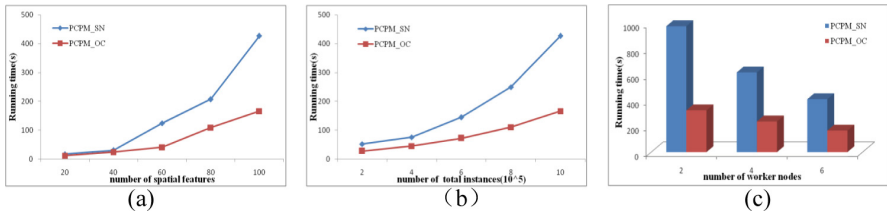


Fig. 5. Running time over synthetic data sets: (a) by the number of spatial features, (b) by the number of spatial instances, (c) by the number of worker nodes

In Fig. 5(c), the performance of speedup is evaluated. The number of spatial features is 100 and the number of spatial instances is 1,000,000. Increasing worker nodes, the running time of two algorithms is reduced obviously, because the mining tasks are allocated to more nodes to execute. In the case of the same number of worker nodes, the performance of the PCPM_OC is better than the PCPM_SN.

6 Conclusions

In this work, we propose a parallel approach for mining co-location patterns from massive spatial data. Each worker node conducts the co-location pattern mining process through expanding ordered cliques level by level. In our method, there is no candidate set generating and clique checking. The experiments show that our approach has a significant improvement in efficiency and has better scalability. Our method is effective for handling massive spatial data, but collecting table instance sets are still time-consuming. Moreover, the issue of load balance is to be explored also. The above questions will be focused on our future researches.

Acknowledgement. This work is supported by the National Natural Science Foundation of China (61472346, 61662086, 61762090), the Natural Science Foundation of Yunnan Province (2015FB114, 2016FA026), and the Project of Innovative Research Team of Yunnan Province.

References

1. 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)
2. Yoo, J.S., Shekhar, S.: A joinless approach for mining spatial colocation patterns. *IEEE Trans. Knowl. Data Eng.* **18**(10), 1323–1337 (2006)
3. Yoo, J.S., Shekhar, S.: A partial join approach for mining co-location patterns. In: *The 12th Annual ACM International Workshop on Geographic Information Systems*, pp. 241–249 (2004)
4. Wang, L., Bao, X., Zhou, L.: Redundancy reduction for prevalent co-location patterns. *IEEE Trans. Knowl. Data Eng.* **30**(1), 142–155 (2018)
5. Xiao, X., Xie, X., Luo, Q., Ma, W.: Density based co-location pattern discovery. In: *16th ACM SIGSPATIAL*, pp. 1–10 (2008)

6. Lin, Z., Lim, S.J.: Fast spatial co-location mining without cliqueness checking. In: International Conference on Information and Knowledge Management, pp. 1461–1462 (2008)
7. Yoo, J.S., Boulware, D., Kimmey, D.: A parallel spatial co-location mining algorithm based on MapReduce. In: IEEE International Congress on Big Data, pp. 25–31 (2014)
8. Wang, L., Bao, X., Chen, H., Cao, L.: Effective lossless condensed representation and discovery of spatial co-location patterns. *Inf. Sci.* **436–437**(2018), 197–213 (2018)