MINING CO-LOCATION PATTERNS FROM SPATIAL DATA

C. Zhou[a,*], W. D. Xiao[a,b], D. Q. Tang[a,b]

[a] Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, 410073 Changsha, China - cheng.zhou@hotmail.com

[b] Collaborative Innovation Center of Geospatial Technology, 430072 Wuhan, China - (wilsonshaw, dqtang)@vip.sina.com

KEY WORDS: Itemset mining, Spatial data , Co-location

ABSTRACT:

Due to the widespread application of geographic information systems (GIS) and GPS technology and the increasingly mature infrastructure for data collection, sharing, and integration, more and more research domains have gained access to high-quality geographic data and created new ways to incorporate spatial information and analysis in various studies. There is an urgent need for effective and efficient methods to extract unknown and unexpected information, e.g., co-location patterns, from spatial datasets of high dimensionality and complexity. A co-location pattern is defined as a subset of spatial items whose instances are often located together in spatial proximity. Current co-location mining algorithms are unable to quantify the spatial proximity of a co-location pattern. We propose a co-location pattern miner aiming to discover co-location patterns in a multidimensional spatial data by measuring the cohesion of a pattern. We present a model to measure the cohesion in an attempt to improve the efficiency of existing methods. The usefulness of our method is demonstrated by applying them on the publicly available spatial data of the city of Antwerp in Belgium. The experimental results show that our method is more efficient than existing methods.

1. INTRODUCTION

With the boom in the availability of spatial data, spatial data mining, i.e., discovering interesting and previously unknown but potentially useful patterns from large spatial datasets, has become a popular field. A co-location pattern represents a subset of spatial items that frequently appear together in spatial proximity. Spatial co-location patterns may yield important insights for many applications, including city planning, mobile commerce, earth science, biology, transportation, etc. For example, location based service providers are very eager to know what services are requested frequently together and located in spatial proximity. This information can help them improve the effectiveness of their location based recommendation system where users request a service in a nearby location and enable the use of pre-fetching to speed up service delivery. Therefore, spatial co-location pattern mining is one of the most crucial spatial data mining tasks.

Figure 1 shows a 2-dimensional structure consisting of four different items, a, b, c, and d. Each item is represented by a distinct shape. The subscript indexes next to the items are only used to identify individual instances of each item, to avoid having to explicitly refer to their coordinates.

A typical approach for mining co-location patterns is proposed by Shekhar and Huang (Shekhar and Huang, 2001, Huang et al., 2004). This method first identifies co-location patterns of size 1 and 2. In our example, given a neighbourhood distance threshold of 100, the approach identifies the neighbourhood relationship of all point pairs, as depicted by solid lines in Figure 2. A clique represents an instance of a set of items located in the same neighbourhood. For example, in Figure 2, $\{a_2, b_3, d_5\}$ is an instance of itemset $\{a, b, d\}$, but $\{a_2, b_3, d_8\}$ is not. Next, the approach generates candidates of size $n + 1$ ($n \geq 2$) and tests the prevalence of each candidate to identify co-location patterns of size $n + 1$. For each candidate of size $n + 1$, the method first joins the instances of two of its subset co-location patterns of size n that share the first $n - 1$ items to identify its instances. The proposed algorithm then computes the prevalence of the candidate

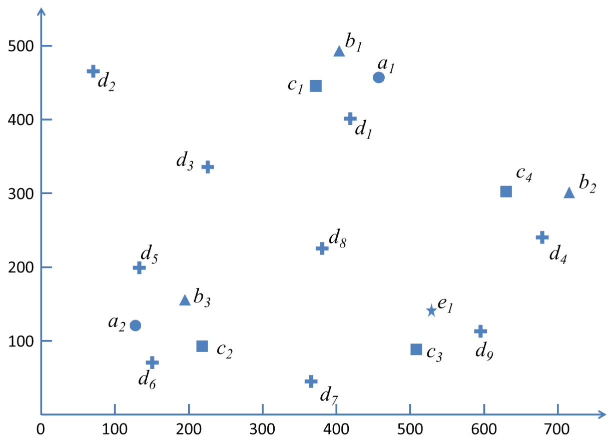

Figure 1. An example of a 2-dimensional structure.

based on the identified instances. If the prevalence of the candidate is not lower than a user-defined prevalence threshold, it is identified as a co-location pattern. The prevalence $prev(P)$ of a pattern P is defined as $prev(P) = \min\{pr(t_i, P), t_i \in P\}$, where $pr(t_i, P)$ is the participation ratio of an item t_i in pattern P, which is defined as

$$pr(t_i, P) = \frac{\text{\# instances of } t_i \text{ in any instance of } P}{\text{\# instances of } t_i}.$$

For example, as shown in Figure 2, $pr(a, \{a, d\}) = \frac{2}{2} = 1$, which reflects the fact that 100% of instances of a (i.e., a_1 and a_2) participate in some instances of pattern $\{a, d\}$ (i.e., $\{a_1, d_1\}$, $\{a_2, d_5\}$ and $\{a_2, d_6\}$). The prevalence of pattern $\{a, b, c\}$ is 0.5, because $pr(a, \{a, b, c\}) = 1$, $pr(b, \{a, b, c\}) = \frac{2}{3}$, and $pr(c, \{a, b, c\}) = 0.5$.

A number of more efficient co-location mining algorithms (Zhang et al., 2004, Yoo and Shekhar, 2006, Xiao et al., 2008) using the same prevalence measure have been proposed afterwards. How-

*Corresponding author

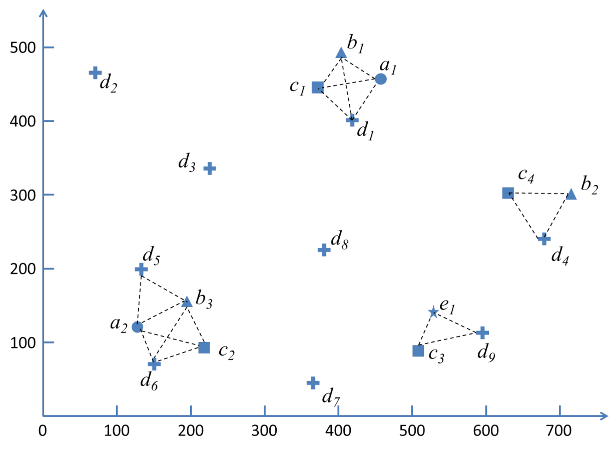

Figure 2. The neighbourhood relationship.

ever, it has been shown that a prevalence threshold based mining approach may fail to find true patterns and may even report meaningless patterns (Barua and Sander, 2014). The prevalence measure value of a set of items can be low, if one participating item has a high participation ratio, but other participating items have low participating ratios due to their large number of occurrences. Overall, the minimum participation value of such patterns will be low and they will be ignored by the existing co-location mining approaches.

Furthermore, the method proposed by Barua and Sander (Barua and Sander, 2014), as well the prevalence based methods (Shekhar and Huang, 2001, Huang et al., 2004, Zhang et al., 2004, Yoo and Shekhar, 2006, Xiao et al., 2008) search for meaningful patterns for a given proximity neighbourhood. Neighbourhood information is given in the form a neighbourhood threshold which is the maximum distance between instances of any two participating items of a pattern. Therefore, there are actually two user-defined thresholds, i.e., a prevalence threshold and a neighbourhood threshold. As a result, a random pattern can attain a high prevalence value, when a large neighbourhood threshold is used by the existing algorithms and be reported as prevalent. A random pattern may also have a high prevalence value with a smaller neighbourhood threshold if the participating items are abundant.

The rest of the paper is organised as follows. Section 2 provides an overview of the existing related work. We formally describe the problem setting for finding co-location patterns in Section 3. In Section 4, we present our algorithm for generating co-location patterns. Section 5 demonstrates the effectiveness of our algorithm applied on the open data of Antwerp and we end the paper with a summary of our conclusions in Section 6.

2. RELATED WORK

Co-location pattern mining approaches are mainly based on spatial relationship such as "close to" proposed by Koperski and Han (Koperski and Han, 1995). Morimoto proposed a method to find groups of various service types originating from nearby locations and reported a group if its frequency of occurrences is above a given threshold (Morimoto, 2001). The basic definitions of co-location pattern mining and the general framework was first proposed by Shekhar and Huang (Shekhar and Huang, 2001). The paper proposes a prevalence measure with an anti-monotonic property, which helps to build an Apriori (Agrawal and Srikant, 1994) based approach to mine prevalent co-location patterns in a reduced search space. This was later extended through the usage of a multi-resolution pruning technique to prune non-prevalent co-locations at a reduced computational cost (Huang et al., 2004). A faster method for the same problem setting was proposed by Zhang et al. (Zhang et al., 2004), while the problem of mining co-location patterns with rare spatial items has been studied by Huang et al. (Huang et al., 2006).

The above methods (Shekhar and Huang, 2001, Huang et al., 2004, Zhang et al., 2004, Huang et al., 2006) have used spatial join approaches to identify patterns whose items are close to each other. A join operation, however, is expensive to compute when searching for patterns. The space required to store all pattern instances is large, and the instance search cost before joining will also increase significantly. To further improve the runtime, Yoo et al. proposed a new join-less approach where a neighbourhood relationship is materialized from a clique type neighbourhood and a star type neighbourhood, respectively (Yoo and Shekhar, 2006). Xiao et al. proposed a density based approach to improve the runtime (Xiao et al., 2008). From a dense region, the method counts the number of instances of a candidate co-location. Assuming all the remaining instances are in co-locations, the method then estimates an upper bound of the prevalence value and if it is below the threshold the candidate co-location is pruned.

However, all the above mentioned methods use a prevalence threshold and look for prevalent patterns based on a user-defined neighbourhood threshold. Therefore, if thresholds are not selected properly, meaningless co-location patterns could be reported, or meaningful co-location patterns could be missed when the prevalence threshold is too high. Barua and Sanders proposed a statistical approach to mine true patterns without using a prevalence measure threshold but this method, too, uses a given neighbourhood threshold (Barua and Sander, 2014).

Afterwards, Zhou et al. proposed a cohesion based co-location pattern miner (Zhou et al., 2015). They measured the spatial proximity of a pattern by defining the cohesive radius of a pattern. The cohesive radius measures the average size of the spatial areas in which the minimal occurrences of the pattern are located. The main benefit of this method is that it allows us to quantify the spatial proximity of a pattern, requiring the user to specify only a cohesive radius threshold. However, the process of computing the cohesive radius is time consuming. In this paper, we propose a new model to improve the efficiency.

3. PROBLEM SETTING

We try to improve the work on mining spatially cohesive itemsets in the spatial data of a city (Zhou et al., 2015), so we begin by adapting some of the necessary definitions from that paper to our setting. We consider an n-dimensional structure S as a set of points where a point v is a pair (t, c) consisting of an item $t \in I$, and an n-dimensional coordinate $c \in \mathbb{R}^n$, where I is the set of all possible items in S and $n \geq 1$. As can be seen in Figure 1, an item t may occur many times at different positions in a structure S. Thus there may be many points containing t in S and we denote such points as V_t, e.g., $V_a = \{a_1, a_2\}$. We denote the frequency of item t as $Fre(t) = |V_t|$. We denote the structure by $S = \{v_1, \ldots, v_l\}$, where $|S| = l$ is the number of points in the structure, i.e., the size of the structure.

Our goal is to investigate patterns of items occurring spatially in close proximity. To do this, we will define co-location patterns in terms of cohesion making it possible to find itemsets consisting of items that, on average, appear close to each other.

3.1 Cohesive Distance

Given a set of points $V = v_1, \ldots, v_q$, let $MiniBall(V)$ denote the ball with the smallest radius that contains V, namely the *smallest enclosing ball*. Zhou et al. considered the points V in n-dimensional space cohesive if the radius of $MiniBall(V)$ is small enough (Zhou et al., 2015). However, we find that the process of finding the smallest enclosing ball costs too much time. Therefore, we propose a new way to measure the *cohesion* of an itemset X in a given structure S. As we know, the diameter is the longest possible chord of any circle. Consequently, if the radius of $MiniBall(V)$ is small enough, then half of the maximum distance between any two points of V will be small enough too. Let $MaxD(V)$ denote the maximum distance between any two points of V. We think the points V in n-dimensional space cohesive if $MaxD(V)$ is small enough.

Given an itemset $X = \{t_1, \ldots, t_m\}$, assume that each item t_i occurs n_i times in structure S. From a point $v_j = (t_i, c_j)$, we can find a smallest $MaxD(V)$, where V contains point v_j and all other items of X, and we denote this smallest $MaxD(V)$ as $d_{v_j}(X)$. Then, we can measure the *cohesion* of an itemset X in a given structure S by computing the average value of $d_{v_j}(X)$s. We call this computed average the *cohesive distance* of X in S.

In order to find all $d_{v_j}(X)$s, we need to check each occurrence of an item in X. In other words, for an item t_i, we need to check n_i times. To do this, for each occurrence of t_i (i.e., v_j) we need to examine each possible combination (i.e., each of the possible combinations of occurrences of all other items in X) in order to find $d_{v_j}(X)$. Repeating this for every item in X, requires finding $m \prod_{i=1}^{m} n_i$ combinations, which is computationally expensive. As a result, we propose a method to approximate this process.

Intuitively, points that occur near to each other are more likely to produce the smallest $MaxD(V)$ than those far apart. Therefore, rather than looking at all possible combinations, we limit our search to a selection of points. We propose an algorithm (as described in Algorithm 1) to first find a cohesive combination $C_v(X)$ containing a point $v = (t_i, c)$ (c is the n-dimensional coordinate of the point and $i \in \{1, \ldots, m\}$) and occurrences of all other items of X nearest to v.

Algorithm 1: Finding a cohesive combination

```
Input  : itemset $X = \{t_1, \ldots, t_m\}$, point $v = (t_i, c)$ with
         $1 \le i \le m$
Output: a cohesive combination $C_v(X)$
1 $C_v(X) = \emptyset$;
2 for $j = 1, \ldots, m$ do
3     $C_v(X) = C_v(X) \cup \arg\min_{w \in V_{t_j}} D(w, v)$
4 return $C_v(X)$;
  // $D(w, v)$ is the Euclidean distance between $w$ and $v$
```

Lemma 1. *Given itemset $X = \{t_1, \ldots, t_m\}$, and a point $v = (t_i, c)$, with $1 \le i \le m$, $MaxD(C_v(X))$ found by Algorithm 1, can never be more than twice as large as $d_v(X)$.*

Proof. Given a data point v, Algorithm 1 will get a combination $C_v(X)$ enclosing v and the occurrences of all other items in X closest to v. Since we know that there exists a combination V with $MaxD(V) = d_v(X)$, we can conclude that, for any item t_j in X, the maximal distance from v to the nearest occurrence of t_j cannot be larger than $d_v(X)$. It follows that $MaxD(C_v(X))$ is at most $2 \times d_v(X)$. □

Based on Lemma 1, we approximate the process of computing the cohesive distance of X as follows:

1. select item t_1 from $X = \{t_1, \ldots, t_m\}$ (items are sorted by frequency in descending order since this computes a more accurate average (Zhou et al., 2015)), and for each point $v_j \in V_{t_1}$, $j = 1, 2, \ldots, |V_{t_1}|$, we find $C_{v_j}(X)$ as described in Algorithm 1 and get $MaxD(C_{v_j}(X))$.

2. we denote the cohesive distance of X in a structure S as

$$D(X) = \frac{\sum_{v_j \in V_{t_1}} MaxD(C_{v_j}(X))}{|V_{t_1}|}. \qquad (1)$$

There are only $|V_{t_1}|$ combinations to find in a structure S, much fewer than if we tried to check all combinations for each occurrence of t_1 in S, resulting in a considerable reduction in time complexity.

However, this procedure inevitably results in approximation errors. For example, as illustrated in Figure 3, assume we are evaluating itemset *abc*, and we picked item a as the first item. We look for the nearest b and the nearest c, and find b_1 and c_1, which are closer to a_1 than b_2 and c_2, respectively, resulting in the dashed line whose length is $MaxD(a_1b_1c_1)$. However, the smallest $MaxD(V)$, where V contains a, b and c, is much smaller, and is depicted using a solid line, i.e., $d_{a_1}(abc)$ = length of a_1c_2. In practice, such cases are rarely encountered. Therefore, this approximate method is capable of producing reasonably accurate results.

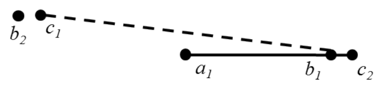

Figure 3. An example of an approximation error.

3.2 Co-location Pattern

Given a maximum cohesive distance threshold *max_dis*, X is a *co-location pattern* or a *cohesive itemset* if $D(X) \le$ *max_dis*. In this case, we say that X is cohesive. Note that the smaller the distance $D(X)$ the higher the cohesion of X. A single item will always be cohesive since the cohesive distance of a singleton is always equal to 0.

The constraint of this approximate method gives a guarantee that when the first item from a co-location pattern is encountered, the remainder of the set is likely to be found nearby.

4. COLOCATION PATTERN MINING ALGORITHM

In this section we present an algorithm for mining co-location patterns in a single structure containing a number of multidimensional points.

4.1 Search For Itemsets

Figure 4 shows the process of enumerating the frequent itemsets, given that the items $\{a, b, c, d\}$ are sorted, e.g., by ascending or descending frequency. Our method enumerates itemsets in a depth-first manner, i.e., we will process itemsets $\{ab, ac, ad\}$, followed by $\{abc, abd, abcd\}$, and finally acd, before moving on to itemsets whose first item is not a.

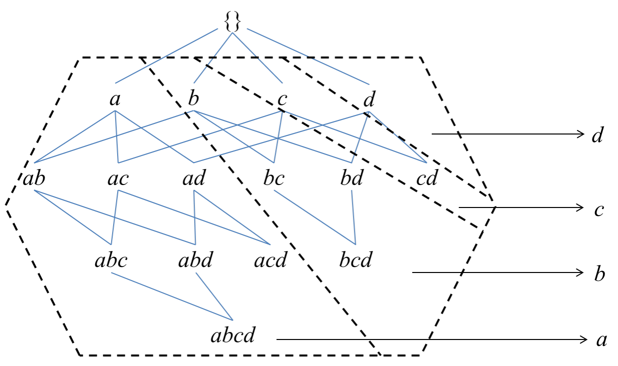

Figure 4. Depth-first search.

4.2 Pruning

The cohesive distance of an itemset is not a monotonic measure. In other words, it is possible for the cohesive distance of a smaller itemset to be greater than the cohesive radius of one of its supersets. For example, $D(bc)$ may turn out to be larger than $D(abc)$ as we are computing the maximum distances around different data points. As shown in Figure 5, $D(bc) > D(abc)$ since the average of $MaxD(b_1c_1)$ and $MaxD(b_2c_1)$ is much larger than $MaxD(a_1b_1c_2)$.

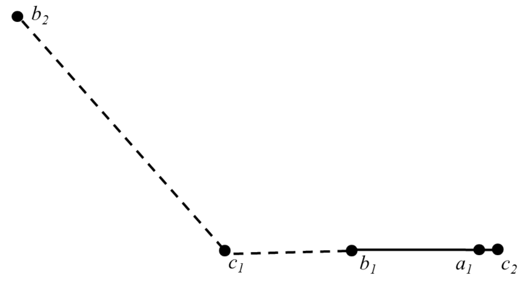

Figure 5. An example resulting in a larger itemset having a smaller cohesive distance.

As a result, we are unable to use the standard itemset mining pruning techniques that rely on the quality measure (typically frequency) being anti-monotonic. However, we here present alternative pruning techniques that are applicable in our method, which allow us to develop an efficient algorithm for the cohesion based co-location pattern mining task.

Although the cohesive distance of an itemset is not monotonic, we can still use its properties for pruning certain candidates from the search space. Our pruning method is based on the following lemma:

Lemma 2. *Assume itemset X is a subset of itemset Y, and all the items of X and Y are sorted by the same order. If they share the same first item, then $D(X) \leq D(Y)$.*

Proof. Denote the first item in X and Y with t. Given an occurrence v_i of t, our method finds a cohesive combination $C_{v_i}(X)$ for itemset X. Clearly, $MaxD(C_{v_i}(Y))$ cannot be smaller than $MaxD(C_{v_i}(X))$ if we insist that $C_{v_i}(Y)$ also contains the nearest occurrences of all items in $Y \setminus X$. Our method finds such combinations $C_{v_i}(Y)$ for each occurrence of t, and then computes the average value of $MaxD(C_{v_i}(Y))$. Given that X and Y share the first item t, the number of such combinations will be the same for X and Y, and each $MaxD(C_{v_i}(Y))$ will be at least as large as the corresponding $MaxD(C_{v_i}(X))$. Therefore, $D(Y)$ will be at least as large as $D(X)$. □

Therefore, we can prune itemsets which are not cohesive when we generate itemsets in the depth-first way as shown in Figure 4.

4.3 Algorithm

After choosing the way of enumerating the itemsets and the pruning method, we design the algorithm to generate all co-location patterns. Our algorithm generates all co-location patterns in two steps. In the first step, we use the depth-first search method to generate candidate itemsets. In the second step, we determine which of the itemsets are actually spatially cohesive and utilise the observations above to prune the itemsets that cannot be cohesive.

Let *n-itemset* denote an itemset of size n. Let F_n denote the set of *n-itemset*s and T_n be the set of cohesive *n-itemset*s. Algorithms 2 and 3 show the process of generating all co-location patterns. Frequency constraints *min_fre* and *max_fre* can be used to filter out items which are not interesting, due to being either too frequent or not frequent enough. For example, in our *city* dataset (see Section 5 for details), there are trash cans on every corner, and patterns including trash cans are therefore of little value to the user. Optional parameters, *min_size* and *max_size*, can be used to limit the output only to co-location patterns with a size bigger than or equal to *min_size* and smaller than or equal to *max_size*.

Algorithm 2: GENERATINGCOLOCATIONPATTERNS. An algorithm for generating all co-location patterns in a structure.

```
Input : structure $S$, frequency constraints $min\_fre$, $max\_fre$,
        maximum cohesive distance threshold $max\_dis$, pattern
        size constraints $min\_size$, $max\_size$.
Output: all co-location patterns $T$.
1 $F_1 = \{t | t \in I, min\_fre \leq Fre(t) \leq max\_fre\}$;
2 if $1 \geq$ min_size then
3     $T_1 = F_1$;
4 sort($F_1$);
5 Depth-First-Search($F_1$);
6 $T = \bigcup T_i$;
7 return $T$;
```

Algorithm 3: Depth-First-Search(Q)

```
Input : a set of itemsets $Q$ sharing all but the last item
1 foreach $\alpha_i$ in $Q$ do
2     $F_i = \emptyset$;
3     foreach $\alpha_j$ in $Q$, with $j > i$ do
4         $X = \alpha_i + last\_item(\alpha_j)$;
5         if $|X| \leq max\_size$ and $D(X) \leq max\_dis$ then
6             $F_i = F_i \cup \{X\}$;
7             if $|X| \geq min\_size$ then
8                 $T_{|X|} = T_{|X|} \cup \{X\}$;
9     Depth-First-Search($F_i$);
```

In Algorithm 2, lines 1-3 count the frequency of all the items to determine the cohesive 1-itemsets. Line 4 sorts the items in F_1 by descending frequency. Line 5 calls Algorithm 3 to get cohesive n-itemsets ($max\_size \geq n \geq 2$). Finally, we get the complete set of co-location patterns T (lines 6-7).

In Algorithm 3, given any two n-itemsets α_i and α_j that share the same first $n-1$ items, we generate a candidate itemset X

of length $n + 1$ by adding the last item in α_j to α_i (line 4). In lines 5-6, we prune the candidates that cannot be cohesive by the properties. Then in lines 7-8, we store the cohesive itemsets into T_n.

5. EXPERIMENTS

We compared our pattern miner, called *CoDis*, with the method proposed by Huang et al. (Huang et al., 2004) (which we call *LW-prev* since the method mines prevalent patterns by a "level-wise" approach) and the *1-descend* method proposed by Zhou et al. (Zhou et al., 2015). We implemented the methods in Java and all experiments were performed on a 2.90GHz Ubuntu machine with 2GB memory.

All the presented methods use some sort of a distance threshold dt, i.e., the maximum cohesive distance threshold *max_dis* for our method *CoDis*, the neighbourhood threshold nt for *LW-prev*, and the maximum cohesive radius threshold *max_rad* for *1-descend*. In all experiments, we keep $dt = nt = max\_rad = \frac{max\_dis}{2}$ to make a fair comparison.

The *city* dataset we used is one 2-dimensional structure obtained from the open data of the city of Antwerp in Belgium[1]. We first downloaded the datasets containing coordinates of different infrastructure objects with locations, e.g., schools, kindergartens, city offices, playgrounds, cultural institutions, public toilets, recycling centres, trash cans, waste recycling bins, glass recycling bins, hospitals, and so on. We expanded the dataset by adding some data about the city neighbourhoods[2] from 2009, i.e., average age, percentage of immigrant population and average income per person, all of which are numeric attributes. Therefore, we first discretised such numbers into different levels based on the information given on the website and used the coordinate of the centroid of a neighbourhood as its location. Table 1 shows a few examples of the items generated in this way. Finally, we merged the datasets together, and thus obtained a 2-dimensional structure containing 6424 points carrying 33 different items.

Attribute	Item Example	Meaning
average age	age42-45	the average age in the neighbourhood is between 42 and 45
percentage of immigrant population	immigrant10-20	the percentage of immigrant population in the neighbourhood is between 10% and 20%
average income per person	income15k-30k	the average annual income (in euros) per person in the neighbourhood is between 15k and 30k

Table 1. Examples of items generated from the demographic data.

5.1 Analysis of Discovered Patterns

The most cohesive itemsets turned out to be singletons, which was to be expected, since singletons always have a cohesive radius equal to 0. To obtain more meaningful results, we decided to look only for itemsets of size 2 or higher. Therefore, for all methods, *min_size* was set to 2 and *max_size* unlimited. We further found that most of the cohesive itemsets contained trash cans, glass recycling bins or recycling bins, which was not surprising, as there were 3750 trash cans, 551 glass recycling bins and 344 recycling bins in the dataset, making their frequencies a lot larger than that of any other item. As a result, we disregarded these items by setting *max_fre* to 340. We set *min_fre* to 5 to prevent items that hardly ever occur from becoming part of a pattern.

We first ran *CoDis* with *max_dis* = 600 metres, discovering the 20 patterns shown in Table 2 in 0.122 seconds. D means the cohesive radius of the discovered co-location pattern and $|V_{t_1}|$ is the frequency of the first item of the pattern. Concrete examples of interesting patterns included the fact that the higher the average age, the higher the average income (patterns 3,6,7 and 18) in a neighbourhood, or that given a kindergarten, there is likely to be a playground nearby (pattern 1). From patterns 2 and 12, we can see that immigrants are likely to be young people. Compare to the patterns mined by *1-descend* with *max_rad* = 300 metres, we find that *CoDis* gets the same patterns.

NO.	Co-location pattern	D	$\lvert V_{t_1}\rvert$
1	kindergarten, playground	335.82	206
2	immigrant10-20, age42-45	419.09	60
3	income<15k, age<36	432.71	104
4	kindergarten, renewal area	452.52	206
5	playground, dog walking area	471.51	184
6	income15k-30k, age42-45	496.49	110
7	income15k-30k, age39-42	504.40	110
8	playground, public toilet	518.35	184
9	income<15k, dog walking area	519.81	104
10	public toilet, income15k-30k	521.64	121
11	renewal area, playground	528.58	191
12	immigrant33-50, age36-39	529.20	53
13	kindergarten, public toilet	541.12	206
14	playground, income15k-30k	544.73	184
15	kindergarten, income15k-30k	553.58	206
16	school, age39-42	573.54	82
17	kindergarten, income<15k	574.60	206
18	income<15k, age36-39	586.34	104
19	kindergarten, dog walking area	586.74	206
20	school, age42-45	596.80	82

Table 2. Co-location patterns found by *CoDis*.

5.2 Impact of Distance Threshold on Runtime

Figure 6 shows the runtimes of the methods with various distance thresholds where other parameters are kept the same as before. We ran *LW-prev* with the prevalence threshold set to 0.1. The results show that, as the distance threshold increases, the runtimes of all methods increase. This is because a larger distance threshold will increase the number of candidate patterns that need to be processed. We find that the runtime of *LW-prev* increases too fast while the runtimes of other methods are acceptable. It can be noted that *LW-prev* runs out of memory when the distance threshold is 800 since the process of identifying prevalent co-location patterns needs more memory than the computations required by our algorithms. *CoDis* appears to be the most scalable method, and the runtime of *CoDis* is almost half of the runtime of *1-descend*.

5.3 Impact of Structure Size on Runtime

Figure 7 shows the runtimes of the methods with respect to different number of points (varying from 10% to 100% of the whole structure). In this experiment we use the whole structure without setting *min_fre* and *max_fre* for the items. We set *min_size* to 2, *max_size* unlimited, and the distance threshold to 500 metres for

[1] http://opendata.antwerpen.be/
[2] http://www.antwerpen.buurtmonitor.be/

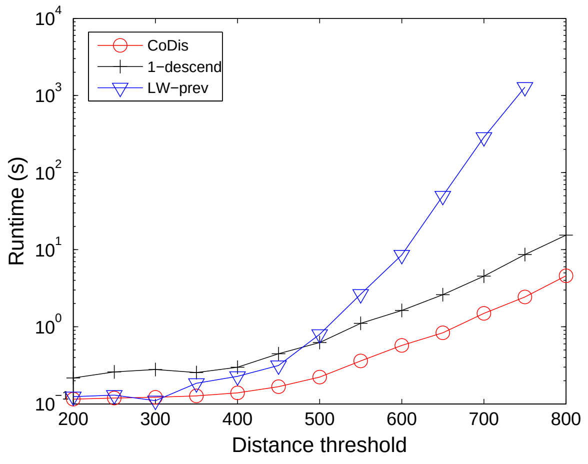

Figure 6. Impact of the distance threshold on the runtime.

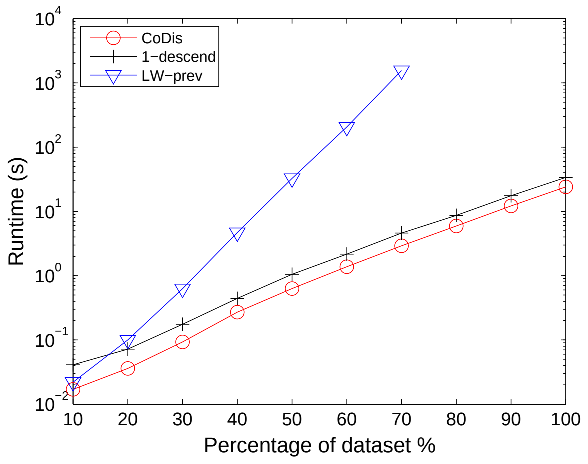

Figure 7. Impact of the structure size on the runtime.

all methods. We ran LW-prev with the prevalence threshold set to 0.1. We repeated the experiment ten times, using ten different random permutations of the data points. The reported runtimes are therefore averages of the ten different runs. For all methods, the runtime grows with increasing number of points. The *LW-prev* method seems to be prohibitive, while *CoDis* is the fastest. Comparing the resulting patterns, we find that *CoDis* performs comparably in terms of accuracy while achieving much quicker runtimes.

6. CONCLUSION

The abundance of spatial data provides exciting opportunities for new research directions but also demands caution in using these data. Handling the very large volume and understanding complex structure in spatial data are two major challenges for spatial data mining, which demand both efficient computational algorithms to mine large datasets for interesting patterns.

In this paper, we have presented a method *CoDis* to efficiently mine co-location patterns in multidimensional spatial data. We applied the method to find spatially cohesive patterns from the spatial data of a city and the resulting patterns demonstrated the efficiency and intuitiveness of the proposed method. Through experimental evaluation, we confirmed that *CoDis* improve the efficiency of *1-descend* by avoiding to find the smallest enclosing ball of points. *CoDis* gives a guarantee that when the first item from a co-location pattern is encountered, the remainder of the set will be found nearby.

REFERENCES

Agrawal, R. and Srikant, R., 1994. Fast algorithms for mining association rules. In: *VLDB'94*, Morgan Kaufmann Publishers, pp. 487–499.

Barua, S. and Sander, J., 2014. Mining statistically significant co-location and segregation patterns. *IEEE Transactions on Knowledge and Data Engineering* 26(5), pp. 1185–1199.

Huang, Y., Pei, J. and Xiong, H., 2006. Mining co-location patterns with rare events from spatial data sets. *Geoinformatica* 10(3), pp. 239–260.

Huang, Y., Shekhar, S. and Xiong, H., 2004. Discovering colocation patterns from spatial data sets: a general approach. *IEEE Transactions on Knowledge and Data Engineering* 16(12), pp. 1472–1485.

Koperski, K. and Han, J., 1995. Discovery of spatial association rules in geographic information databases. In: *Advances in spatial databases*, Springer, pp. 47–66.

Morimoto, Y., 2001. Mining frequent neighboring class sets in spatial databases. In: *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, pp. 353–358.

Shekhar, S. and Huang, Y., 2001. Discovering spatial co-location patterns: A summary of results. In: *Advances in Spatial and Temporal Databases*, Springer, pp. 236–256.

Xiao, X., Xie, X., Luo, Q. and Ma, W.-Y., 2008. Density based co-location pattern discovery. In: *Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems*, ACM, p. 29.

Yoo, J. S. and Shekhar, S., 2006. A joinless approach for mining spatial colocation patterns. *IEEE Transactions on Knowledge and Data Engineering* 18(10), pp. 1323–1337.

Zhang, X., Mamoulis, N., Cheung, D. W. and Shou, Y., 2004. Fast mining of spatial collocations. In: *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, pp. 384–393.

Zhou, C., Cule, B. and Goethals, B., 2015. Cohesion based co-location pattern mining. In: *Proceedings of the International Conference on Data Science and Advanced Analytics (DSAA).*

ACKNOWLEDGEMENTS

We would like to thank the government of Antwerp for making the data of Antwerp publicly available.