Mining Spatial Co-location Patterns with Dynamic Neighborhood Constraint

Feng Qian, Qinming He, and Jiangfeng He

College of Computer Science and
Technology, Zhejiang University, Hangzhou, China
{qfeng,hqm,jerson_hjf2003}@zju.edu.cn

Abstract. Spatial co-location pattern mining is an interesting and important issue in spatial data mining area which discovers the subsets of features whose events are frequently located together in geographic space. However, previous research literatures for mining co-location patterns assume a static neighborhood constraint that apparently introduces many drawbacks. In this paper, we conclude the preferences that algorithms rely on when making decisions for mining co-location patterns with dynamic neighborhood constraint. Based on this, we define the mining task as an optimization problem and propose a greedy algorithm for mining co-location patterns with dynamic neighborhood constraint. The experimental evaluation on a real world data set shows that our algorithm has a better capability than the previous approach on finding co-location patterns together with the consideration of the distribution of data set.

Keywords: Spatial data mining, Spatial co-location patterns, Spatial neighborhood constraint.

1 Introduction

Spatial co-location pattern mining [1] is an interesting and important issue in spatial data mining area which discovers the subsets of features (co-locations) whose events are frequently located together in geographic space. Its application domains include M-commerce, earth science, biology, public health, transportation, etc [2]. Take M-commerce as an example, in mobile computing, to provide location based services, service providers may tend to find services requested frequently located in spatial proximity.

Previous research literatures [1,2,3,4,5,6,7,8,9] for mining spatial co-location patterns require two constraints as prerequisite: the neighborhood constraint for spatial events and the prevalence constraint for spatial feature sets. As a result, two corresponding thresholds should be given from domain experts to help describe the constraints. A typical mining approach is carried out by two steps: identify the neighbor relations of spatial events with the neighborhood constraint; calculate the values of prevalence measure for different co-location candidates based on the neighbor relations of their events and find the winners whose values meet the prevalence constraint satisfaction. Apparently, some drawbacks are introduced by fixing on the neighbor relations early.

W. Buntine et al. (Eds.): ECML PKDD 2009, Part II, LNAI 5782, pp. 238–253, 2009.

1. It's hard to choose a definitely appropriate neighborhood threshold, even for domain experts. Usually, some repetitive trials should be put into practice to investigate the data sets.
2. The relationship between the neighbor relations and the prevalence measure is simply split.
3. Event types have different distributions in different areas of the global space. A static neighborhood threshold is too simple to evaluate their neighbor relations.

Hence, we suggest in this paper a heuristic approach which postpones the determination of neighbor relations to the process of the second step. A greedy algorithm is proposed to carefully select neighbor relations in stead of determining them through a uniform comparison with a simple distance based threshold. The selection strategy is based on the preferences of finding nontrivial co-locations early and the dynamic configuration of the spatial framework. Therefore, in our study of mining spatial co-location patterns, the neighborhood constraint is dynamic.

Related Work. The first general framework of mining spatial co-location patterns was proposed by Huang et al. [1] which adopts the aforementioned two-step approach. After that, different algorithms were proposed to improve the efficiency of the mining process, such as partial-join algorithm [3] and join-less algorithm [2] proposed by Yoo et al., and density based algorithm [4] proposed by Xiao et al.. In these literatures, the neighborhood constraint is described by a distance threshold which is the maximal distance allowed for two events to be neighbors. The work was extended to complex spatial co-location patterns by Munro et al. [5] and Verhein et al [6]. All these researches are limited to Huang's framework and approach.

Other researches deal with the interest measure for the prevalence of spatial co-location patterns. Huang et al. [7] adjusted the measure to treat the case with rare events. There was no change with the neighborhood constraint. Another algorithm was also given by Huang [8] that uses density ratio of different features to describe the neighborhood constraint together with a clustering approach. A buffer based model was used to describe the neighborhood constraint by Xiong et al. [9] that deal with extended spatial object such as lines and polygons. However, in the above algorithms, once the neighborhood constraint is given by user, the neighbor relations are determined and never change during the later process of evaluating the prevalence of co-locations.

On the other hand, in the field of spatial statistics, hypothesis testing methods are used to identify the correlation patterns. In Salmenkivi's work [10], the neighbor relations and prevalence measure were bound together. However, the work is not cost effective and furthermore can not handle the case of a co-location with more than two features. Sheng et al. [11] introduced the definition of influence function based on gaussian kernel to describe the neighborhood constraint. However, the algorithm assumed a distribution of features on the global space and it is expensive to compute the value of influence function.

Contributions. In this paper, we make the following contributions:

1. We conclude the preferences for selecting neighbor relations that are qualified for finding co-location patterns on early stage.
2. We define the mining task as an optimization problem and propose a greedy algorithm to mine co-location patterns with dynamic neighborhood constraint.
3. We also experimentally evaluate the algorithm on a real world data set and compare it with the previous approach.

The remainder of the paper is organized as follows: In Section 2, the relationship between neighborhood constraint and prevalence constraint is investigated together with several observations of the preferences for determining qualified neighbor relations for finding co-locations early. Section 3 define the co-location mining task as an optimization problem. A greedy algorithm for mining co-location patterns with dynamic neighborhood constraint is proposed in Section 4. Section 5 presents the experimental evaluation. And conclusion is discussed in Section 6.

2 Relationship between Neighborhood Constraint and Prevalence Constraint

In this section, we investigate the relationship between neighborhood constraint and prevalence constraint based on Huang's framework [1].The qualification of neighbor relations for detecting co-locations and the preferences for selecting qualified neighbor relations are studied. We first present some basic concepts.

Given a set of spatial features F, a set of their events E, and a neighborhood constraint, the objective of spatial co-location pattern mining is to find co-location rules in the form of $C_1 \rightarrow C_2$ $(p,\ cp)$, where C_1, $C_2 \subseteq F$, and $C_1 \cap C_2 = \emptyset$ [2]. We denote the **co-location** as C=$\{C_1 \cup C_2\}$. The interest of a co-location rule can be measured by the constraints of prevalence (p) and conditional probability (cp).

The prevalence constraint is described as **participation index** (Pi) in the framework which is $Pi(C)$=$min_{f_i \in C}\{Pr(C, f_i)\}$ where $f_i \in F$ and Pr is **participation ratio** defined as $Pr(C,f_i)$=$\frac{Number\ of\ distinct\ events\ of\ f_i\ in\ instances\ of\ C}{Number\ of\ events\ of\ f_i}$ [2]. Since the conditional probability is similar defined and can be straightly measured based on prevalence constraint, we skip the discussion of it in the paper. We call an instance of C as a **clique instance** $(CI(C))$ [1] that all events in it are neighbors to each other based on the neighborhood constraint. In Huang's framework, the neighborhood constraint is described by a distance threshold which is the maximal distance allowed for two events to be neighbors.

To investigate the relationship between neighborhood constraint and prevalence constraint, we study an US National Transportation Atlas Database with Intermodal Terminal Facilities (NTAD-ITF) [12]. Every event in the data set has a location information with latitude and longitude, and is a facility with

Table 1. Description of NTAD-ITF with attributes

No.	Attribute	Description
1	ID	Event id.
2	TYPE	Name of the function of the primary function of the facility.
3	LATITUDE	Latitude for the location of the facility.
4	LONGITUDE	Longitude for the location of the facility.

Table 2. Description of NTAD-ITF with event number of facility types

No.	Facility Type	Abbreviation	Number of events
1	RAIL	R	2157
2	AIRPORT	A	398
3	TRACK	T	443
4	PORT	P	172
5	INTER PORT	I	87

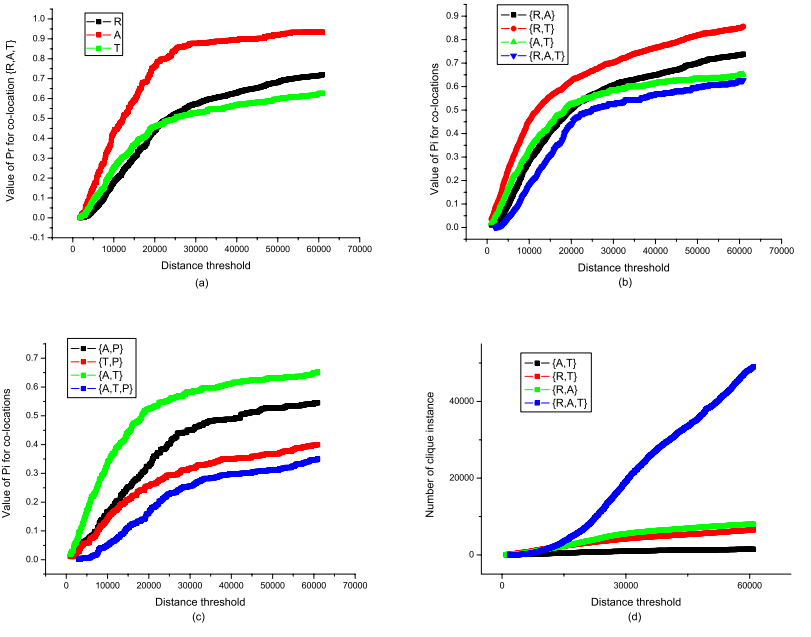
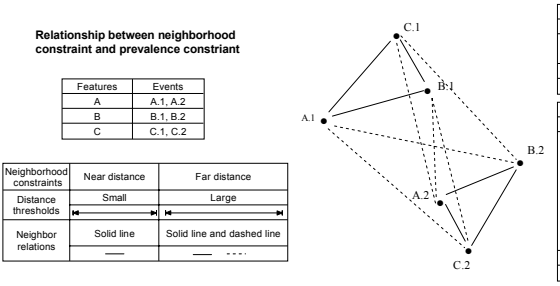


Fig. 1. The value of participation ratio (a), participation index (b,c) and the number of clique instances for co-locations (d) along with the distance threshold



Parameters of co-locations with small distance threshold

Co-locations	{A,B}	{A,C}	{B,C}	{A,B,C}
Clique instances	A1,B1	A1,C1	B1,C1	A1,B1,C1
	A2,B2	A2,C2	B2,C2	A2,B2,C2
Pr	2/2,2/2	2/2,2/2	2/2,2/2	2/2,2/2,2/2
Pi	1	1	1	1

Parameters of co-locations with large distance threshold

Co-locations	{A,B}	{A,C}	{B,C}	{A,B,C}
Clique Instances (Appended instances with grey color)	A1,B1	A1,C1	B1,C1	A1,B1,C1
	A2,B2	A2,C2	B2,C2	A2,B2,C2
	A1,B2	A1,C2	B1,C2	A1,B1,C2
	A2,B1	A2,C1	B2,C1	A1,B2,C1
				A1,B2,C2
				A2,B1,C1
				A2,B1,C2
				A2,B2,C1
Pr	2/2,2/2	2/2,2/2	2/2,2/2	2/2,2/2,2/2
Pi	1	1	1	1

Fig. 2. An example to illustrate the relationship between neighborhood constraint and prevalence constraint

different types such as rail, airport, track, port and inter port. We applied the join-less algorithm [2] on the data set. The co-located relationships among these types are investigated. The geographic coordinates were transferred to projection coordinates using Universal Transverse Mercartor (UTM) projection. The detailed information about the data set is described in table 1 and table 2.

As the result of the experiments, we plotted the values of participation ratio for co-location {R,A,T} (with abbreviation described in table 2) with distance threshold (in units of miles) in figure 1(a). It is obvious that the values of participation ratio increased fast with small distance threshold. When the distance threshold reached around 20000, the increasing became slow. Figure 1(b,c) plotted the similar behavior for values of participation index along with distance threshold for different co-locations and their size 2 subsets: {R,A,T} and {A,T,P}. Moreover, {R,A,T} had a tighter bound to its subsets than that of {A,T,P}. Figure 1(d) plotted the number of clique instances with distance threshold for co-locations as figure 1(b). As can be seen from the figure, the number of clique instances for {R,A,T} increased slowly for small distance threshold compared with its subsets. But after the distance threshold reaching around 12000, the number increased faster and quickly exceeded the others.

Figure 2 simply illustrates the reason why the curves behave that way in figure 1. In the figure, each event is uniquely identified by $T.i$, where T is the feature type and i is the unique id inside each feature type. For example, A.2 represents the second event of feature A. We first describe the neighborhood constraint by a small distance threshold which confirms the neighbor relations by the solid lines. In the right-top table of figure 2, we enumerate the clique instances of the co-locations as well as the value of participation ratio and participation index. Despite of a small distance threshold, participation index reach the value of one. When it comes to a large distance threshold, the neighbor relations are confirmed by the solid lines and the dashed lines together. As can be seen from the right-bottom table in figure 2, the number of clique instances increases from 2 to 4 for size 2 co-locations and from 2 to 8 for size 3 co-locations. That is why in figure 1 (d) the long co-location run faster than the short co-locations as distance threshold increased. Because long co-locations have exponentially more

combinations of clique instances than short co-locations based on the same set of events. On the contrary, the appended clique instances benefitting from the increasing of distance threshold contribute much less (nothing in the example) to the value of participation ratio as well as participation index. That is why the curves in figure 1 (a,b,c) increased slowly for large distance threshold.

Based on the analysis, we conclude the preferences for selecting qualified neighbor relations that are in favor of finding nontrivial co-locations.

1. Near distance neighbor relations are preferred.
2. Neighbor relations that benefit the value of prevalence measure are preferred.
3. Co-locations whose values of prevalence measure keep tight to that of their subsets are preferred.
4. A big gap of numbers of clique instances between co-locations and their subsets implies too much noise.

The first preference is obvious due to Tobler's first law of geography: everything is related to everything else but nearby things are more related than distance things [13]. Since the algorithms tend to find nontrivial co-locations that stand for high values of prevalence measure on early stage, the second preference promises. We prove preference 3 by comparing {R,A,T} and {A,T,P} with figure 1 (b,c). Recall the apriori approach [14] adopted by almost all of the algorithms for mining co-location patterns presently. The size k candidate co-locations are generated from their size k-1 subset co-locations as well as their instances. If these candidates are pruned a lot in the validation process or even in the earlier generated process, their values of prevalence measure leave far away from that of their size k-1 subsets, as to other shorter subsets. Hence, as can be seen from the figure 1 (b,c), domain experts may prefer {R,A,T} to {A,T,P} as a nontrivial co-location since it had a tighter bound value of prevalence measure on its subsets. The fourth observation may not be a preference but a condition the algorithm is supposed to stop its trial for those co-locations, since it imposes that the new generated clique instances benefit from far neighbor relations and they also contribute less to the value of prevalence measure.

3 Co-location Pattern Mining as an Optimization Problem

Based on the observations we made on the preferences that algorithms rely on for finding nontrivial co-locations, we can intuitively define the mining task as an optimization problem. The problem aims to discover on early stage as much as co-locations with high values of prevalence measure and near distance neighbor relations for their instances.

Specifically, we describe the task into a placement problem. Initially, the events in the spatial framework have no neighbor relation with each other. Then the neighbor relations are placed back in order to the framework. Once any neighbor relation is placed, the configuration of the framework is refreshed together

with the reward of the placement. We measure the reward of the placement by the benefits of values of prevalence measure of the co-locations over the distances of neighbor relations as follows.

$$R(\alpha) = \sum_{C \subseteq F} (\theta^{-k} \cdot f(C) \cdot t(C) \cdot \sum_{d_j \in N_C} \frac{\Delta_{d_j} Pi(C)}{|d_j|^{\omega}}) \tag{1}$$

In equation 1, α is a placement and F represents the set of spatial features. Every candidate co-location C is first weighted by the step function f that is defined following (k > 2).

$$f(C_k) = \begin{cases} 1 & \text{if } Pi(C_k) \geq (\eta \cdot E[Pi(C_{k-1})]) \; where \; C_{k-1} \subset C_k \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Here, C_k is a size k co-location and $E[Pi(C_k)]$ is the average value of prevalence measure for the co-locations. The function simply prunes the size k co-locations whose values of prevalence measure leave far away from that of their size k-1 subsets. The coefficient η is used to adjust the extent of value difference which is between 0 and 1. It is obvious that algorithms only consider the benefit value as reward when the absolute value of prevalence measure is high enough. This corresponds to the preference 2 mentioned before.

In equation 1, function t weights the value difference of size k co-locations with their size k-1 subsets as described following (k > 2). It corresponds to the third preference mentioned in the last section. Due to the monotonic property of participation index [1], the value of function t is between 0 and 1.

$$t(C) = \frac{Pi(C_k)}{E[Pi(C_{k-1})]} \; where \; C_{k-1} \subset C_k \tag{3}$$

In equation 1, N_C represents the set of neighbor relations that participate in the instances of co-location C. For each neighbor relation, we calculate the incremental values of prevalence measure of co-locations that benefit from placing it to the framework. The distance of neighbor relation d_j is a penalty to the value in the equation which corresponds to the first preference. ω is a coefficient emphasizing the weight of distance to the benefit value which is usually between 0 and 1. And $\Delta_{d_j} Pi(C)$ is the incremental value of $Pi(C)$ from placing d_j into the current framework. Note, the neighbor relations are placed in selected order. Therefore the values are accumulated to evaluate the reward of placement. θ is a coefficient to adjust the user's bias on long co-locations which is between 0 and 1. In the equation, k is the size of the co-location C. The estimations for both function f and t could be skipped for size 2 co-locations since there are not comparable values of prevalence measure for them with size 1 co-locations.

As described above, the co-location mining task can be made into a optimization problem and the greedy algorithm can be applied to maximize the marginal gain when select neighbor relations for spatial framework.

$$d_i = \underset{d_i \in D \setminus \alpha_{i-1}}{\operatorname{argmax}} \; R(d_i) = \underset{d_i \in D \setminus \alpha_{i-1}}{\operatorname{argmax}} \; R(\alpha_{i-1} \cup d_i) - R(\alpha_{i-1}) \tag{4}$$

Here, D is the set of neighbor relations and d_i is the neighbor relation the greedy algorithm selects in the ith iteration based on *i-1*th configuration. To make the algorithm feasible, we simplify the evaluation of neighbor relations in each iteration as follows.

$$R(d_i) = \frac{1}{|d_i|^{\omega}} \cdot \sum_{C \subseteq F} (\theta^{-k} \cdot f_{i-1}(C) \cdot t_{i-1}(C) \cdot \Delta_i Pi(C)) \tag{5}$$

The deformation of function f and function t indicates that the values of the functions are calculated based on *i-1*th configuration of the spatial framework.

Recall the fourth preference presented in section 2. The ratio of the average number of clique instances of a co-location's subsets over the number of clique instances of itself could give us a signal that whether the evaluation of the co-location should be stopped. As a constraint to this optimization problem, we can simply implement this by comparing the value of ratio with a threshold κ. If the value is below κ, the evaluation of the co-location should be skipped in the iterations (k > 2) as presented following.

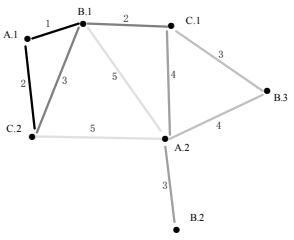
$$Evaluate(C_k) = \begin{cases} 1 & \text{if } \frac{E[|CI(C_{k-1})|]}{|CI(C_k)|} \geq \kappa \text{ where } C_{k-1} \subset C_k \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

The estimation can be easily embedded into equation 1 and equation 5 as a multiplication factor. If all the co-locations reach the line, the algorithm terminates. Other estimations can be applied in a similar way, such as defining a maximum value for the prevalence measure. Actually, as to this heuristic approach, the user can pause the estimation at any time and check the values of prevalence measure. Then, the algorithm could go on from the pausing point.

4 A Greedy Algorithm for Co-location Pattern Mining

Thus, a greedy algorithm could be carried out naturally. However, the problem is still not submodular [15] due to the possible incremental benefits for the remaining unplaced neighbor relations after refreshing the configuration when some neighbor relations were selected into the framework. Hence, the lazy evaluation approach proposed by Leskovec et al. [15] can not be adopted to speed up the algorithm.

Despite this, we propose to select a set of neighbor relations at each placing instead of a single one. We also propose to evaluate the neighbor relations in a qualified buffer instead of all the neighbor relations. The intuition comes from the thinking that we are not actually interested in maximizing the final reward of the placement but to observe as much as nontrivial co-locations. Especially on early stage, too many re-evaluations exhaust the computation, whereas a precise rank of the qualified neighbor relations is unnecessary since they will eventually take part in the early placement. Oppositely, it's also not necessary for unqualified neighbor relations to attend in early evaluations.



Features	Events	Coefficients	Values
A	A.1, A.2	θ	1
B	B.1, B.2, B.3	η	0
C	C.1, C.2	ω	1

Note 1:
1. Weights on the lines indicate neighbor distances;
2. Deepness of lines indicates the order of placement;
3. Other wasted neighbor relations are ignored in the figure.

Rank of neighbor relations for sequential configurations

Rank 0	Disance	Rank 1	R(d1)	Rank 2	R(d2)
A.1,B.1	1	**A.1,B.1**	**0.33**	A.1,B.1	X
A.1,C.2	2	**A.1,C.2**	**0.25**	A.1,C.2	X
B.1,C.1	2	B.1,C.1	0.17	**B.1,C.2**	**0.22**
A.2,B.2	3	A.2,B.2	0.11	**B.1,C.1**	**0.17**
B.1,C.2	3	B.1,C.2	0.11	A.2,B.2	0.11
B.3,C.1	3	B.3,C.1	X	B.3,C.1	0.11
A.2,B.3	4	A.2,B.3	X	A.2,B.3	0.08
A.2,C.1	4	A.2,C.1	X	A.2,C.1	X
A.2,B.1	5	A.2,B.1	X	A.2,B.1	X
A.2,C.2	5	A.2,C.2	X	A.2,C.2	X

Rank of neighbor relations for sequential configurations

Rank 3	R(d3)	Rank 4	R(d4)	Rank 5	R(d5)
A.1,B.1	X	A.1,B.1	X	A.1,B.1	X
A.1,C.2	X	A.1,C.2	X	A.1,C.2	X
B.1,C.2	X	B.1,C.2	X	B.1,C.2	X
B.1,C.1	X	B.1,C.1	X	B.1,C.1	X
A.2,C.1	**0.13**	A.2,C.1	X	A.2,C.1	X
A.2,B.2	**0.11**	A.2,B.2	X	A.2,B.2	X
B.3,C.1	0.11	**B.3,C.1**	**0.11**	B.3,C.1	X
A.2,B.3	0.08	**A.2,B.3**	**0.08**	A.2,B.3	X
A.2,B.1	0	A.2,B.1	0	**A.2,B.1**	**0**
A.2,C.2	X	A.2,C.2	0	**A.2,C.2**	**0**

Clique Instances after sequentially placing neighbor relations

Selection	N.R.	{A,B}	{A,C}	{B,C}	{A,B,C}
1	A.1,B.1	A1,B1			
	A.1,C.2		A.1,C.2		
2	B.1,C.2			B.1,C.2	A.1,B.1,C.2
	B.1,C.1			B.1,C.1	
3	A.2,C.1		A.2,C.1		
	A.2,B.2	A.2,B.2			
4	B.3,C.1			B.3,C.1	
	A.2,B.3	A.2,B.3			A.2,B.3,C.1
5	A.2,B.1	A.2,B.1			A.2,B.1,C.1
	A.2,C.2		A.2,C.2		A.2,B.1,C.2

Rewards of sequential placements

Placement	{A,B}	{A,C}	{B,C}	{A,B,C}	Reward
1	1/3	1/2	0	0	0.55
2	1/3	1/2	1/3	1/3	0.95
3	2/3	1	1/3	1/3	1.19
4	1	1	2/3	2/3	1.38
5	1	1	2/3	2/3	1.38

Note 2:
1. Buffer size = 5, selection size = 2;
2. Purple indicates rank buffer in reward order;
3. Green indicates neighbor relations not in the buffer but in distance order;
4. Bule indicates clique instances genereated by neighbor ralations inside a selection of placing.

Fig. 3. An example to illustrate processes of greedy algorithm for placement problem

An example is illustrated in figure 3 where we select 2 neighbor relations at each placing step and the buffer size is 5. Initially, the neighbor relations are sorted in distance order which coarsely estimates their qualifications. Then we fill the buffer with the first 5 neighbor relations in distance order. The evaluation is carried out on the buffer and the neighbor relations inside the buffer are sorted by reward order.

The rewards are calculated following the equation 5 where the values of coefficients are described in figure. Take {B.1,C.1} as an example, its reward can be calculated by dividing the benefit of $Pi(\{B,C\})$ by its distance which is (1/3)/2 = 0.17. We select the first 2 neighbor relations in the ranked buffer to place into the framework. For example, after the first evaluation, the neighbor relations {A.1,B.2} and {A.1,C.2} are selected into the framework since they gain the top rewards of 0.33 and 0.25. After that, the statistics for the clique instances together with the values of prevalence measure are updated.

In the next iteration, the first 2 neighbor relations outside the buffer but in distance order are chosen to make up the buffer. In the example, they are {B.3,C.1} and {A.2,B.3}. Then the similar process is implemented in buffer. As we stated above, possible incremental benefits may be introduced for some neighbor relations. In the example, since the placed neighbor relations {A.1,B.2} and {A.1,C.2} could enhance {B.1,C.2} with potential benefit of $Pi(\{A,B,C\})$, its reward increases and rank position arises. Therefore, iteration by iteration, sets of qualified neighbor relations are selected into the framework and the algorithm goes on.

However, it's unreasonable to calculate the final reward by simply accumulating the evaluated rewards of the selected neighbor relations. The estimated reward of a selection can be effected by two opposite cases. In the first case, some neighbor relations in the same selection may share the same benefit. For example, {B.1,C.2} and {B.1,C.1} share the same benefit of $Pi(\{B,C\})$ with value of 1/3 in the second selection. On the contrary, in the second case, some

clique instances could be generated by neighbor relations inside a selection which introduces more benefits. In the above example, {A.2,B.3,C.1} is generated by {B.3,C.1} and {A.2,B.3} inside the fourth selection. Hence, a compromise should be made to meet both the cases. We give the estimation of the reward for a selection as follows.

$$R(S) = \frac{\sum_{di \in S} R(di) \cdot |di|^{\omega}}{E[|di|^{\omega}]} \tag{7}$$

The estimation equation calculates the weighted mean for the evaluated reward of each neighbor relation. In the equation, S is the set of neighbor relations within a selection. And $\mathrm{E}[|di|^{\omega}]$ is the average value of $|di|^{\omega}$ where di is a neighbor relation within a selection that $di \in S$. As an example in figure 3, the estimated reward of the first selection can be calculated by adding the weighted rewards of {A.1,B.1} and {A.1,C.2}, then dividing the sum by their average distance which is $(0.33 \times 1 + 0.25 \times 2)/(1.5) = 0.55$. Therefore, the reward of the placement can be calculated by accumulating the estimated rewards of the selections.

Actually, there's no need to know all the distances of neighbor relations in advance. We only need to keep a set of neighbor relations with the number of buffer size at each iteration. Hence, two operations are required: initially find the first *bsize* shortest neighbor relations for buffer where *bsize* the buffer size; at each iteration find the first *ssize* shortest neighbor relations in the rest neighbor relations where *ssize* is the selection size. The detailed algorithms are described following.

Algorithm 1. Find the first *bsize* shortest neighbor relations for buffer

```
1 for every event e in framework do
2 |   e' = find_distinct_nearest_neighbor(e);
3 |   insert_into_queue(make_neighbor_relation(e, e'));
4 sort_queue();
5 while size of buffer < bsize do
6 |   append_to_buffer(find_next_shortest_neighbor_relation());
7 return buffer;
```

Algorithm 1 first find the nearest neighbor for each event in the spatial framework and insert the neighbor relation into a queue. The event is called the reference event of the neighbor relation. If the found neighbor relation is duplicated with an existed one in the queue, it is discarded and the next nearest neighbor for this event is pursued until a distinct neighbor relation is found (***Line 1-3***). The process of finding the nearest neighbor can be implemented by plane sweeping [16]. After that, the queue is sorted in distance order and at each iteration the shortest neighbor relation outside the buffer is found through invoking method in algorithm 2 and appended to the buffer. The iteration goes on until the buffer is full (***Line 4-6***). Finally, the buffer is returned (***Line 7***).

Algorithm 2. Find the next shortest neighbor relation

```
1 d = pop_top_neighbor_relation_in_queue();
2 e' = find_next_distinct_nearest_neighbor(reference event e of d);
3 insert_into_queue_by_order(make_neighbor_relation(e, e'));
4 return d;
```

Algorithm 2 first pop out the top neighbor relation in the sorted queue (***Line 1***). For this neighbor relation, the next distinct nearest neighbor of its reference event is found. This neighbor relation should not be duplicated with those found before (***Line 2***). Then the neighbor relation is inserted into the queue by distance order and finally the answer is returned (***Line 3-4***). The second operation can be implemented by invoking algorithm 2 *ssize* times.

Next, we draw the greedy algorithm for mining co-location patterns following.

Algorithm 3. The greedy algorithm for mining co-location patterns

```
1 init_buffer(bsize);
2 while not reach termination conditions do
3 |   evaluate_buffer(θ, η, ω);
4 |   sort_buffer_by_reward_order();
5 |   pop_top_neighbor_relations_in_buffer_to_framework(ssize);
6 |   update_configuration();
7 |   make_up_buffer(ssize);
8 return co-locatins whose prevalence value above prevalence threshold;
```

Algorithm 3 first initializes the buffer by invoking the method of algorithm 1 (***Line 1***). Then the iteration starts by evaluating the buffer. Every neighbor relation in the buffer is evaluated following equation 5 together with a set of coefficients (***Line 3***). Then, the neighbor relations in the buffer are sorted in the evaluated reward order (***Line 4***). The top *ssize* neighbor relations are popped out and placed into the spatial framework (***Line 5***). After that, the algorithm updates the statistical information for the configuration of the spatial framework which includes the updated clique instances, the values of prevalence measure and the accumulated reward of the current placement (***Line 6***). By invoking the method of algorithm 2, the next *ssize* shortest neighbor relations are found to make up the buffer (***Line 7***). Finally, when termination conditions are met, the co-locations whose prevalence value above the prevalence threshold are returned (***Line 2,8***).

In the process of evaluating neighbor relations, the benefits of values of prevalence measure are calculated by first finding new clique instances introduced by the evaluated neighbor relation. The size 2 clique instances are naturally ready. Besides, the join approach [1] is used to generate longer clique instances. We skip the detailed explanation to save the space.

Table 3. Experimental parameters and their values in experiments

Parameters	Description	Ex1	Ex2	Ex3	Ex4	Ex5
θ	A coefficient in equation 1.	0.2				
η	A coefficient in equation 2.	0.05				
ω	A coefficient in equation 1.	0.2				
bsize	Size of buffer.	1000,1500,2000			500-2000	1000
ssize	Size of selection.	75,113,150			100	100-400
sratio	Selection ratio from buffer.	0.075			*	

5 Experimental Evaluation

In this section, we evaluate the greedy algorithm on a real world data set that is a US National Transportation Atlas Database with Intermodal Terminal Facilities (NTAD-ITF) [12]. Detailed information about the data set is described in table 1 and table 2 in section 2. The co-located relationships among the 5 types of facilities are investigated. A previous efficient algorithm (join-less algorithm [2]) based on Huang's framework [1] is also evaluated as a comparison.

The algorithms are implemented in C++ and are memory based algorithm. All the experiments were performed on an Intel Core 2 Duo 2.2GHz E4500 machine with 2G MB main memory, running Windows XP operation system. Detailed information about experiments is listed in table 3.

5.1 Capability of the Algorithms for Finding Co-location Patterns

We first evaluated the capability of the algorithms for finding co-location patterns. The capability can be specified in four ways following.

1. Detecting the high values of prevalence measure for co-locations with less neighbor relations participated in the algorithm.
2. Detecting the high values of prevalence measure for co-locations with less clique instances involved in the algorithm.
3. Finding the co-location patterns considering the distribution of the data set.

We first compared the capability of our greedy algorithm and the join-less algorithm with the number of neighbor relations and the number of clique instances. The join-less algorithm was evaluated with several repetitive executions. In each repetition, the distance threshold was incremented with a uniform value. We also compared the algorithms with the average distance of the neighbor relations which partially reflects the distribution of the neighbor relations that the algorithms selected. Detailed information is described in table 3 (Ex1-Ex3).

Capability with the number of neighbor relations. In the first experiment, we evaluated the prevalence values of the co-locations {R,A} and {R,A,T} with the number of neighbor relations as illustrated in figure 4(a) and figure 4(b) respectively. In the figure, different curves represent different algorithms. We plotted three curves of the greedy algorithm with different buffer sizes but the same

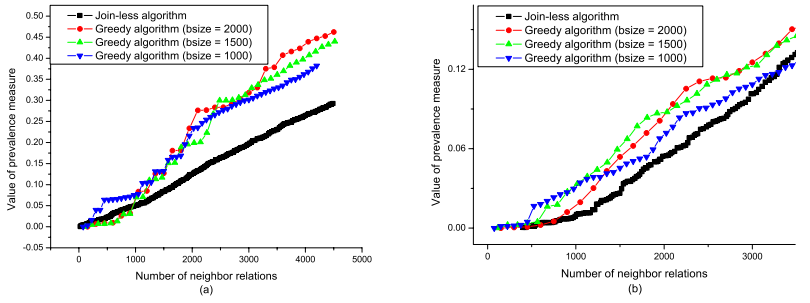


Fig. 4. The value of prevalence measure with the number of neighbor relations for co-location {R,A} (a) and co-location {R,A,T} (b)

selection ratio for the buffer. As can be seen from the figure, the greedy algorithm detected the higher value of prevalence measure with less neighbor relations involved in the calculation due to its preferences of selection. The algorithm tended to select neighbor relations in the density areas of the spatial framework on early stage, since they could benefit more values of prevalence measure. For co-location {R,A,T}, when the number of neighbor relations increased after 3000, the value difference of the two algorithms became smaller. That's because when too many neighbor relations participated in the calculation, many noises as presented in section 2 were also introduced. Then the greedy algorithm tended to select neighbor relations outside the former density areas. However on early stage, the greedy algorithm is capable of finding the co-locations with high values of prevalence measure than join-less algorithm. When buffer size became larger, the greedy algorithm had a better capability, since more qualified neighbor relations were participated in the evaluations which refined the selections.

Capability with the number of clique instances. In the second experiment, we evaluated the prevalence values of the co-locations {R,A} and {R,A,T} with the number of clique instances as illustrated in figure 5(a) and figure 5(b) respectively. As can be seen from the figure, the greedy algorithm detected the higher value of prevalence measure with very less clique instances, since the greedy algorithm selected the neighbor relations into the spatial framework that benefit the values of prevalence measure instead of many noise neighbor relations that could introduce the clique instances whose events have already participated in the calculation of the participation ratios.

Capability with the average distance of the neighbor relations. In the third experiment, we evaluated the number of neighbor relations with the average distance of the neighbor relations as illustrated in figure 6. As can be seen from the figure, the average distance of the greedy algorithm increased faster than join-less algorithm along with the number of neighbor relations. Because in the selection process of the greedy algorithm, the greedy algorithm not only considered the distances of the neighbor relations, but also took into account their quality of detecting high value of prevalence measure which potentially imposes

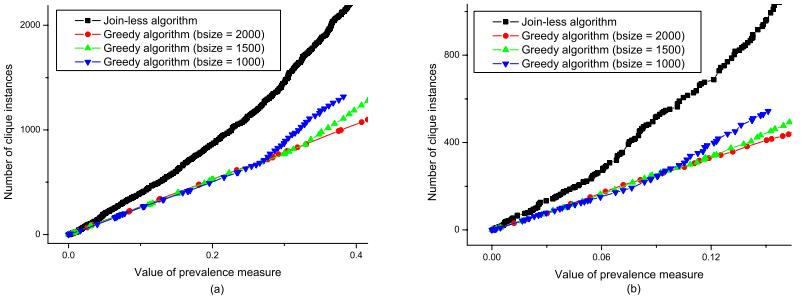


Fig. 5. The number of clique instances with the value of prevalence measure for co-location {R,A} (a) and co-location {R,A,T} (b)

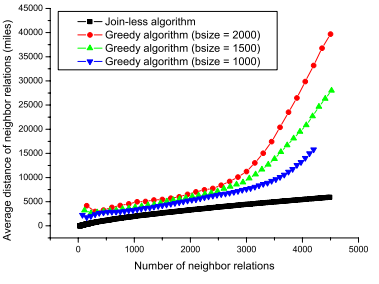


Fig. 6. The average distance of neighbor relations with number of neighbor relations

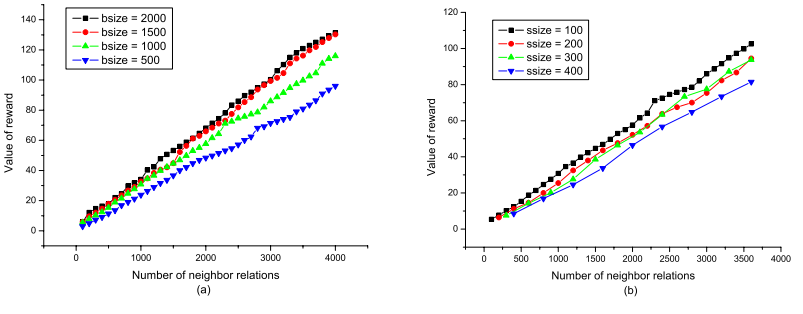


Fig. 7. The value of accumulated reward with the number of neighbor relations for different buffer sizes (a) and different selection sizes (b)

the nontrivial co-locations. Thus, from another point of view, the greedy algorithm tended to select the neighbor relations in density areas of the spatial framework, sine they could introduce more clique instances together with more benefit values of prevalence measure. Therefore, our algorithm consider the distribution of the data set than the previous approach.

5.2 Effects of the Buffer Size and the Selection Size

We evaluated the effects of the buffer size and the selection size to the greedy algorithm. The reward of the placement was used as a reference. The values of the rewards were calculated according to equation 7. Detailed information is described in table 3 (Ex4-Ex5).

Effect of the buffer size. In the fifth experiment, we evaluated the effect of the buffer size with the accumulated reward as illustrated in figure 8(a). We fixed the selection size with 100 neighbor relations in the experiment. As can be seen from the figure, algorithm with larger buffer size derived higher value of reward, since more qualified neighbor relations participated in the evaluation and rank. When the buffer size increased from 1500 to 2000, the incremental value of the reward became small due to more unqualified neighbor relations involved in the rank. That's the reason why the buffer strategy is adopted instead of evaluating all the neighbor relations.

Effect of the selection size. In the sixth experiment, we evaluated the effect of the selection size with the accumulated reward as illustrated in figure 8(b). We fixed the buffer size with 1000 neighbor relations in the experiment. The greedy algorithm with smaller selection size derived higher value of reward, since less unqualified neighbor relations were involved in the evaluation. When the number of neighbor relations increased, the value difference became larger due to the increasing of the number of unqualified neighbor relations.

We skip the discussion of the performance of the algorithms to save the space. We also skip the discussion of the effects of the coefficients (θ, η, and ω) to the greedy algorithm to save the space.

6 Conclusion

In this paper, we first proposed the drawbacks introduced by assuming a static neighborhood constraint for mining co-location patterns which adopted by previous research literatures. Then, we concluded the preferences that algorithms rely on when making decisions for mining co-location patterns with dynamic neighborhood constraint. Based on this, we defined the mining task as an optimization problem and propose a greedy algorithm for mining. The experimental evaluation on a real world data set shows that our algorithm has a better capability than previous algorithms on finding co-location patterns together with the consideration of the distribution of data set. The algorithm is also flexible for other prevalence constraints besides the participation index [1], such as the constraint with density ratio [8] and the constraint with buffer overlap [9].

References

1. Huang, Y., Shekhar, S., Xiong, H.: Discovering colocation patterns from spatial datasets: A general approach. IEEE Transactions on Knowledge and Data Engineering 16(12), 1472–1485 (2004)
2. Yoo, J., Shekhar, S.: A Joinless Approach for Mining Spatial Colocation Patterns. IEEE Transactions on Knowledge and Data Engineering 18(10), 1323–1337 (2006)
3. Yoo, J., Shekhar, S., Smith, J., Kumquat, J.: A partial join approach for mining co-location patterns. In: Proceedings of the 12th annual ACM international workshop on geographic information systems, pp. 241–249 (2004)
4. Xiao, X., Xie, X., Luo, Q., Ma, W.: Density based co-location pattern discovery. In: Proceedings of the 16th ACM SIGSPATIAL international conference on advances in geographic information systems (2008)
5. Munro, R., Chawla, S., Sun, P.: Complex spatial relationships. In: Proceedings of the 3th IEEE international conference on data mining, pp. 227–234 (2003)
6. Verhein, F., Al-Naymat, G.: Fast Mining of Complex Spatial Co-location Patterns Using GLIMIT. In: Proceedings of the 7th international conference on data mining workshop on spatial and spatio-temporal data mining, pp. 679–684 (2007)
7. Huang, Y., Pei, J., Xiong, H.: Mining Co-Location Patterns with Rare Events from Spatial Data Sets. GeoInformatica 10(3), 239–260 (2006)
8. Huang, Y., Zhang, P., Zhang, C.: On the Relationships between Clustering and Spatial Co-location Pattern Mining. International Journal on Artificial Intelligence Tools 17(1), 55 (2008)
9. Soung, X.: A framework for discovering co-location patterns in data sets with extended spatial objects. In: Proceedings of the 4th SIAM international conference on data mining, p. 78 (2004)
10. Salmenkivi, M.: Evaluating attraction in spatial point patterns with an application in the field of cultural history. In: Proceedings of the 4th IEEE international conference on data mining, pp. 511–514 (2004)
11. Sheng, C., Hsu, W., Li Lee, M., Tung, A.: Discovering Spatial Interaction Patterns. In: Haritsa, J.R., Kotagiri, R., Pudi, V. (eds.) DASFAA 2008. LNCS, vol. 4947, pp. 95–109. Springer, Heidelberg (2008)
12. Bureau of Transportation Statistics, `http://www.bts.gov/`
13. Tobler, W.: Cellular geography. Philosophy in geography, pp. 379–386 (1979)
14. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: Proceedings of the 20th international conference on very large data bases, vol. 1215, pp. 487–499 (1994)
15. Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., VanBriesen, J., Glance, N.: Cost-effective outbreak detection in networks. In: Proceedings of the 13th ACM SIGKDD international conference on knowledge discovery and data mining, pp. 420–429 (2007)
16. Arge, L., Procopiuc, O., Ramaswamy, S., Suel, T., Vitter, J.: Scalable sweeping-based spatial join. In: Proceedings of the 24th international conference on very large data bases, pp. 570–581 (1998)