THE NATIONAL UNIVERSITY
of SINGAPORE

School of Computing
Computing 1, Singapore 117590

TRC6/07

Discovering Spatial Interaction Patterns

Chang SHENG, Wynne HSU
and Mong Li LEE

June 2007

Technical Report

Foreword

This technical report contains a research paper, development or tutorial article, which has been submitted for publication in a journal or for consideration by the commissioning organization. The report represents the ideas of its author, and should not be taken as the official views of the School or the University. Any discussion of the content of the report should be sent to the author, at the address shown on the cover.

OOI Beng Chin
Dean of School

Discovering Spatial Interaction Patterns

Chang Sheng Wynne Hsu Mong Li Lee
School of Computing, National University of Singapore
{shengcha, whsu, leeml}@comp.nus.edu.sg

Abstract

Advances in sensing and satellite technologies and the growth of Internet have resulted in a vast amount of uncertain spatial data. Extracting interaction patterns from these uncertain data is a challenging task. In this paper, we propose to model the spatial features in a continuous space through the use of influence functions. For each feature type, we build an influence map that captures the distribution of the feature instances. Superimposing the influence maps allows the interaction of the feature types to be quickly determined. We design an efficient algorithm called PROBER to find the maximal spatial interaction patterns. Experiments on both synthetic and real world datasets indicate that the proposed approach is scalable and is able to discover patterns that have been missed by existing methods.

1 Introduction

Among the various spatial analysis tasks, finding interaction among spatial features is an important problem with many scientific applications. For example, in epidemiology studies, dengue fever and Aedes mosquito tend to exhibit spatial correlation while in ecology, Nile crocodile and Egyptian plover are often found in tandem. Knowing the set of E-services that are located together is beneficial to mobile companies to improve their location-based services. The analysis of web log can also reveal the interests of customers in different geographical locations.

In view of this, there has been sustained interest in developing techniques to discover spatial collocation patterns. The interestingness measure of these patterns is a *binary* notion of proximity where a *distance threshold* is set and objects are either close or far away based on the threshold. Ripley's K function [8, 2] supports this measure by computing the probability of how objects of one feature is close to objects of one or more features. Interesting collocation patterns are then defined to be a set of features that are frequently close to each other [4, 15].

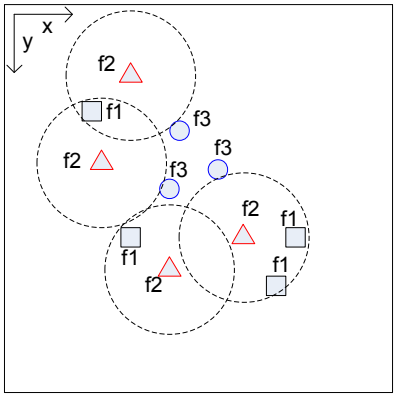
Geo-spatial data are however by nature imprecise due to various reasons which include the limitation of measuring instruments, human recording errors, concern for privacy and dynamic movement of some objects. This means that no since precise point can be used to represent the location of these objects.

Existing collocation mining algorithms, however, does not lend itself easily for handling uncertain spatial data. To address this problem, in this paper, we will model the error distribution of the spatial data to be *Gaussian* instead of a precise point. Figure 1(a) gives a sample dataset of objects [1] with three feature types f_1, f_2, f_3, denoted by the symbols $\square$, $\triangle$, and $\bigcirc$, respectively. Figure 1(b) gives the objects' distribution where the dotted circles define the object's distance threshold. We observe that many of the

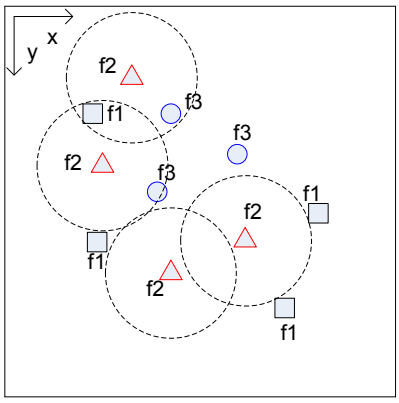
[1] Without special presentation, objects refer to the spatial instances in dataset in the rest of this report.

ID	X	Y	FID
001	22.8	27.5	f_1
002	32.2	60.8	f_1
003	75.1	60.4	f_1
004	70.3	72.5	f_1
005	32.4	17.7	f_2
006	25.1	40.2	f_2
007	42.4	67.0	f_2
008	61.0	61.2	f_2
009	45.2	33.0	f_3
010	54.9	37.0	f_3
011	40.1	48.2	f_3

(a) Dataset



(b) Observed Instance Distribution



(c) Underlying Instance Distribution

Figure 1. Some instances and their spatial relationship

△ objects have □ objects within the distance threshold. In other words, {□, △} is a collocation pattern. However, if the exact locations of the □ instances are as shown in Figure 1(c), then {□, △} will not be a collocation pattern since many of their instance pairs are now outside the distance threshold. At the same time, many ◯ objects are now within the distance threshold of the △ objects.

The above example demonstrates that existing collocation mining approaches, which employ the exact support measure (i.e. the number of instances), are sensitive to the assigned distance threshold. With different distance thresholds, these algorithms may find different collocation patterns. In addition, they do not show good scalability as collocation mining procedure is, in essence, nothing more than the expensive spatial join among multiple datasets [15].

While one may build an uncertain model to capture the underlying distribution for each object, and derive the probability of an object being close to another object for a given distance threshold, such an approach is computationally expensive as each feature may have multiple objects.

Motivated by these challenges, we propose to model the spatial features in a continuous space using the radial basis functions. This approach resembles the kernel density estimation (KDE) in statistics except that KDE does not consider the positional error of the uncertain data. In this paper, we use two Gaussian functions, namely, the error function and the kernel function to model the observed position of the object. The actual influence is computed as the convolution of these two functions. By summing up the influences of all the instances of a feature, we obtain the influence distribution (or the influence map) of the feature.

We introduce the notion of *Spatial Interaction Patterns* (SIPs) to capture the interactions among sets of features. These patterns are sets of binary features whose influence maps are commonly correlated. For each feature type, we build an influence map that captures the distribution of the feature instances. Superimposing the influence maps allows the interaction of the feature types to be easily determined without costly spatial joins. Experiments are performed on both synthetic and real world datasets to demonstrate that the proposed approach is not only efficient but is able to discover patterns that have been missed by existing methods.

The remainder of this paper is organized as follows. Section 2 presents the related work in the area of collocation pattern mining and kernel density estimation. In Section 3, we describe the influence model and its properties. Section 4 introduces the proposed mining algorithm called PROBER. We study the performance of the mining algorithm in Section 5 and summarize our finding in Section 6.

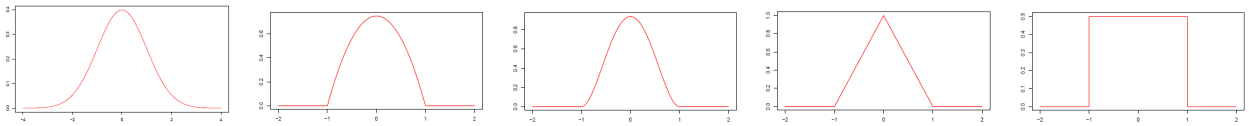
2 Related Works

2.1 Collocation Pattern Mining

In the data mining community, extension from the classic itemset mining framework [1] has led to the concept of spatial association rules [6]. The solution proposed is based on a reference feature centric model which converts the spatial relationship to transactions, thus transforming the problem into association rule mining from transactions. In contrast, other works [4, 7] propose an event centric model whose statistical foundation is based on Ripley's K function [8, 2]. In the event centric model, an *instance* of a pattern P is a set of objects that satisfy the unary (feature) and binary (neighborhood) constraints specified by the pattern's graph. For example, a_1, b_1, c_1 is an instance support the clique pattern $P = \{a, b, c\}$, if only the distance of any two instances is not more than given threshold σ. Two measures, participation ratio and prevalence, are developed to evaluate a candidate pattern.

The main drawback of the event centric model is its requirement for the expensive and frequent instance spatial-join operations during mining. Yoo et.al [14, 13] propose partial-join and join-less approaches to overcome this drawback. The key idea of the algorithms is to enumerate and sort all the neighboring instances into a projected database during the preprocessing stage. Subsequent mining algorithm deals only with the projected database. Similarly, Wang et.al [12] propose the use of a summary structure to store the necessary information for subsequent mining. Zhang et.al [15] utilize a grid partitioning approach to map each feature instance into a grid and perform independent mining on each grid.

All the above works regard the proximity relation as a binary relation. We classify these works as using the *distance model*. Our work, however, views the proximity relation as a continuous function.



(a) Gaussian (b) Epanechnikov (c) Biweight (d) Triangle (e) Uniform

Figure 2. Kernel Functions

2.2 Kernel Density Estimation

The kernel density estimation (KDE) is a statistical method for estimating probability densities without distribution assumptions [11]. The kernel functions model the probability density of the influence of one point in its neighborhood. Contribution from each point is summed to provide an overall estimate. Usually kernel functions are chosen to be unimodal and symmetric about zero. The common kernel functions are Gaussian, Epanechnikov, Biweight, Triangle and Uniform (see Figure 2). Formally, given a kernel function K and a bandwidth h, the kernel density estimator is defined to be

$$\hat{f}_n(x) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{h} K(\frac{x - X_i}{h}),$$

where n is the number of points and bandwidth h controls the amount of smoothing. It has been shown that the choice of kernel K is not crucial, but the choice of bandwidth h is very important [11].

KDE, however, can not be utilized directly to find the spatial interaction, because the bandwidth h of KDE implies different semantics to the spatial affinity in our work. Also, KDE focuses on the density estimation of one feature, while spatial interaction in our work focuses on multiple features.

3 Influence Model

In this section, we introduce the notations used in defining the influence function to capture the degree of affinity between two spatial objects. We extend the notations to the influence maps of features, i.e., object groups, and infer some useful properties.

3.1 Object-to-Object Influence Function

Recall that most spatial data are inherently uncertain with an error distribution modeled by the Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$,

$$e(\mu, x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{|\mu - x|^2}{2\sigma^2}}, \tag{1}$$

where μ is the observed i-th dimensional value and x is the underlying value.

When two spatial objects are near each other, they exert an influence on each other. This degree of influence is represented as a radial basis kernel function in the form of either Gaussian, Epanechnikov, Biweight or Triangle function. Let $k_i(\cdot)$ and $e_i(\cdot)$ denote the kernel function and error function, respectively, on the i-th dimension. Taking into account the effect of error distribution due to the uncertain spatial data, the actual influence exerted between two spatial objects along the i-th dimension is the convolution of $k_i(\cdot)$ with $e_i(\cdot)$, $k_i \otimes e_i$.

Definition 1 *The actual influence of an object on a point in the i-th dimension, denoted as inf_i, is defined as*

$$inf_i = k_i \otimes e_i \equiv \int_{-\infty}^{\infty} e_i(x) k_i(x - \tau) d\tau \tag{2}$$

where $k_i(\cdot)$ is the kernel function and $e_i(\cdot)$ is the error function, along the i-th dimension .

In this work, we select the Gaussian function to be the kernel due to its unique influence range $(-\infty, \infty)$ among candidate kernels. Hence, the influence of an object on a point is the convolution of two Gaussian functions, namely Gaussian kernel and Gaussian error. We know that the convolution of two Gaussians is also a Gaussian function. Without loss of generality, let $e_i = \mathcal{N}(0, \sigma_e^2)$ and $k_i = \mathcal{N}(0, \sigma_k^2)$, $inf_i = k_i \otimes e_i = \mathcal{N}(0, \sigma_e^2 + \sigma_k^2)$.

From this definition, we can easily generalize the influence function to the high dimensional space.

Definition 2 *Assuming that each dimension has identical Gaussian error $\mathcal{N}(0, \sigma_e^2)$ and Gaussian kernel $\mathcal{N}(0, \sigma_k^2)$. Given a d-dimensional object $o = (h_1, h_2, \ldots, h_d)$, its **influence** to the neighboring point $p = (p_1, p_2, \ldots, p_d)$ in the d-dimensional space, is the product of influence on each dimension:*

$$inf(o, p) = \prod_{i=1}^{d} \frac{1}{\sqrt{2\pi(\sigma_e^2 + \sigma_k^2)}} exp\{-\frac{(h_i - p_i)^2}{2(\sigma_e^2 + \sigma_k^2)}\} = (2\pi(\sigma_e^2 + \sigma_k^2))^{-d/2} exp\{-\frac{\sum_{i=1}^{d}(h_i - p_i)^2}{2(\sigma_e^2 + \sigma_k^2)}\} \tag{3}$$

In fact, the exponential factor $\sum_{i=1}^{d}(h_i - p_i)^2$ is the square of Euclidean distance of o and p in d-dimensional space, $(Euclidean(o, p))^2$. We observe that this influence function has the following properties. *Monotonicity*: It is anti-monotonic to the Euclidean distance between two objects in a high dimensional space; and *Robustness*: It takes into consideration the uncertainty in the data.

In the case of a 2D spatial object o on the x-y plane, its influence distribution is a bivariate Gaussian function whose mean is the observed position of o and standard deviation is $\sqrt{(\sigma_e^2 + \sigma_k^2)}$. If the kernel on each dimension has the same standard deviation (i.e., $\sigma_x = \sigma_y$), the influence distribution is circular in shape, otherwise, it is an ellipse. Figure 6 illustrate an influence function of circular shape. We call this bell-like 3D shape an **influence unit**. For the remainder of this paper, we use $\mathcal{N}(0, \sigma^2)$ to denote an influence unit, where $\sigma = \sqrt{\sigma_e^2 + \sigma_k^2}$. Note that the rectangular region denotes a range of mean $\pm 3\sigma$ along the x-y plane and it captures over 95% of the influence exerted by the object.

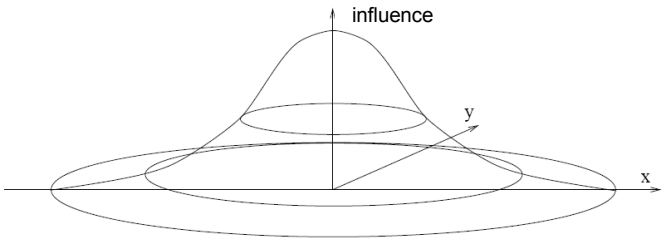
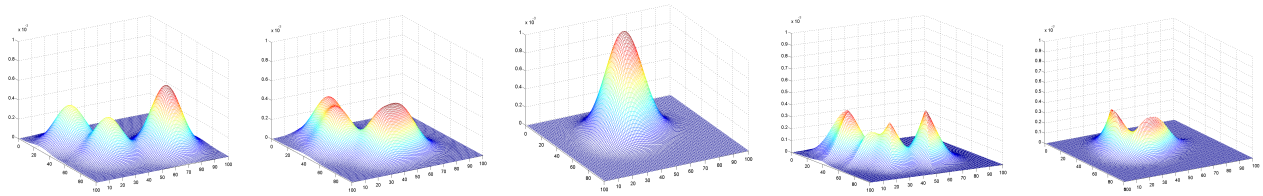


Figure 3. Influence distribution on 2D space

Lemma 1 *The influence measure is symmetric, i.e.,* $inf(o_i, o_j) = inf(o_j, o_i)$.

PROOF: It can be inferred from the Definition 2 as the $Euclidean\ (o_i, o_j) = Euclidean(o_j, o_i)$.□

3.2 Feature-to-Feature Influence Function



(a) $S(f_1)$ (b) $S(f_2)$ (c) $S(f_3)$ (d) $S(f_1, f_2)$ (e) $S(f_2, f_3)$

Figure 4. Examples of influence maps and their interaction

Similar to the kernel density estimator [10], the influence map of a feature on a 2D plane is the normalized summation of all the instances' influence units of this feature [2].

Definition 3 *Given a set of spatial objects* $\{o_1, o_2, \ldots, o_n\}$ *of feature* f *on a spatial plane* $\mathcal{P}$*, and the influence function* $inf(\cdot)$*. The* **influence** *of feature* f *on a position* $p \in \mathcal{P}$*, denoted by* $S(f, p)$*, is*

$$S(f, p) = \frac{1}{n} \sum_{i=1}^{n} inf(o_i, p). \tag{4}$$

We use $S(f)$ *to denote the* **influence map** *of feature* f*.*

[2]Nevertheless, the smoothing techniques well-developed in kernel density estimator are beyond the scope of this paper, because influence maps are only utilized to model the distribution of influences of objects instead of objects themselves

The volume of $S(f)$ can be computed as the integral of the influence of all points in $\mathcal{P}$. This leads to Lemma 2.

Lemma 2 *The volume of an influence map $S(f)$ is 1.*

PROOF: Assume the feature f contains n objects $\{o_1, o_2, \ldots, o_n\}$.
Volume of $S(f) = \int_{p\in\mathcal{P}} S(f,p)dp$
$= \int_{p\in\mathcal{P}} \frac{1}{n}\sum_{i=1}^{n} inf(o_i,p)dp$
$= \frac{1}{n}\int_{p\in\mathcal{P}} inf(o_1,p)dp + \ldots + \int_{p\in P} inf(o_n,p)dp$
$= \frac{1}{n} \times n \times (Volume\ of\ influence\ unit)$
$= 1$. □

Definition 4 *Given two influence maps $S(f_1)$ and $S(f_2)$ with respect to feature f_1 and f_2 and the spatial plane $\mathcal{P}$, the* **influence of a feature pair** *$\{f_1, f_2\}$ on a position $p \in \mathcal{P}$, denoted as $S(f_1, f_2, p)$, is $min\{S(f_1,p), S(f_2,p)\}$. We use $S(f_1, f_2)$ to denote the* **influence map of the feature pair** *$\{f_1, f_2\}$ on plane $\mathcal{P}$.*

Definition 5 *The interaction between a pair of features $\{f_i, f_j\}$ is measured as the volume of the influence map $S(f_i, f_j)$. We call this measure the* **Interaction (I)** *of feature pair $\{f_i, f_j\}$ and is denoted as $I(f_i, f_j) = \int_{p\in P} S(f_i, f_j, p)dp$.*

From Lemma 2, we infer that $0 \leq I \leq 1$. The interaction between a feature and itself is 1, i.e., $I(f_i, f_i) = 1$. Hence, $I(f_i, f_j) = 1$ indicates that the objects of feature f_i and f_j have the same distribution. On the other hand, $I(f_i, f_j) = 0$ implies that the data distributions of feature f_i and f_j are far apart from each other.

Figure 4 shows the influence maps for feature f_1 and f_3, and the feature pair $\{f_1, f_3\}$. Note that Definition 4 and Definition 5 can be easily extended to three or more features. For three features, we have $I(f_1, f_2, f_3) = \int_{p\in\mathcal{P}} min\{S(f_1,p),\ S(f_2,p),\ S(f_3,p)\}dp$. The Interaction measure (I) is used to determine the significance of a spatial interaction. This measure indicates how much a feature is affected by the interaction from the other features in a feature set. Lemma 3 and Lemma 4 gives some important properties of interaction measure.

Lemma 3 *(Symmetry property) The interaction measure is Symmetric, i.e., $I(f_i, f_j) = I(f_j, f_i)$.*

PROOF: It can be inferred from the Lemma 1 and Definition 5 as both $Euclidean()$ and $min()$ subfunctions are symmetric. □

Lemma 4 *(Apriori property) The Interaction measure (I) is monotonically non-increasing as the increase of features.*

PROOF: Let us assume that a feature set P_n consists of n features, $f_0, f_1, \ldots, f_{n-1}$, and the interaction of n features, $f_0, f_1, \ldots, f_{n-1}$, to be $I(f_0, f_1, \ldots, f_{n-1})$. According to Definition 4, $I(f_0, f_1, \ldots, f_{n-1}) = \int_{p\in\mathcal{P}} min\{NS(f_0,p), S(f_1,p), \ldots, S(f_{n-1},p)\}dp$. Then for a longer feature set $P_{n+1} = P \bigcup \{f_n\}$, we have

$I(f_0, f_1, \ldots, f_{n-1}, f_n)$
$= \int_{p\in\mathcal{P}} min\{S(f_0,p), S(f_1,p), \ldots, S(f_{n-1},p), S(f_n,p)\}dp$
$= \int_{p\in\mathcal{P}} min\{min\{S(f_0,p), \ldots, S(f_{n-1},p)\}, S(f_n,p)\}dp$
$\leq \int_{p\in P} min\{S(f_0,p), S(f_1,p), \ldots, S(f_{n-1},p)\}\}dp$
$= I(f_0, f_1, \ldots, f_{n-1})$. □

Lemma 3 implies that we can construct an undirected graph where each node indicates a feature and the edges associated with node pairs indicate the interaction. Lemma 4 implies that the pattern generation may follow the classic Apriori property, avoiding some patterns which are impossible to be valid.

4 Mining Spatial Interaction Patterns

In this section, we present the algorithm PROBER (sPatial inteRactiOn Based pattErns mineR) to find the interaction patterns in spatial databases.

Definition 6 *Given a spatial database containing a feature set $\mathcal{F}$ and a threshold min_I, a* **Spatial Interaction Pattern** *SIP is the set of features $\{f_1, f_2, \ldots, f_k\} \subseteq \mathcal{F}$ and $I(f_i) \geq min\_I, 1 \leq i \leq k$.*

If the interaction of a SIP is greater than a predefined threshold min_I, we call this SIP to be a *valid* SIP or *frequent* SIP. The *maximal* SIP set is the smallest set of valid SIPs that covers all valid SIPs. For example, given the valid SIPs $\{\{f_1, f_2\}, \{f_2, f_3\}, \{f_1, f_3\}, \{f_3, f_4\}, \{f_1, f_2, f_3\}\}$, the maximal SIP set is $\{\{f_3, f_4\}, \{f_1, f_2, f_3\}\}$. Each member in maximal SIP set is called a maximal SIP.

The problem of mining spatial interaction patterns is defined as follows. *Given a spatial database containing m features and n instances, as well as the minimal interaction measure min_I, our goal is to find the set of all valid spatial interaction patterns.*

Mining of SIP is computationally expensive due to two-fold reasons. First, the comparison of continuous spaces is computationally infinite. Second, the enumeration of all candidate patterns is exponential. We consider the first problem in Section 4.1 and examine the second problem in Section 4.2 and 4.3. In Section 4.4, we present the PROBER algorithm and analyze its complexity.

4.1 Uniform Sampling Approximation

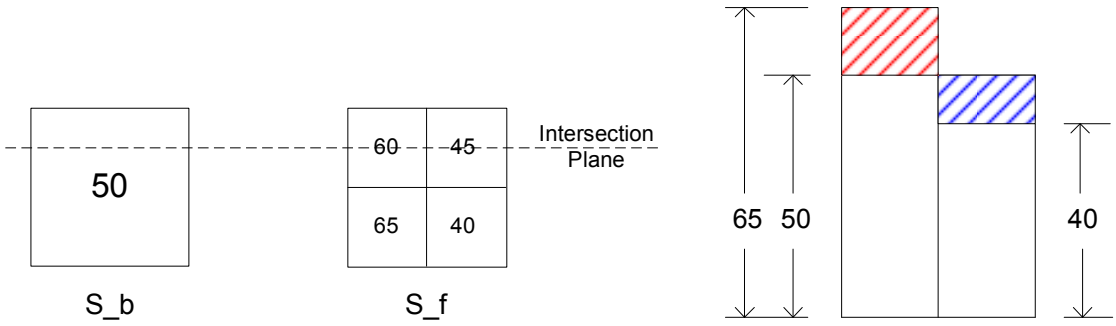
In theory, the influence map of one feature is continuous, which means that there are infinite comparisons when considering the relationship between two influence maps. To expedite the mining process, we uniformly divide the spatial plane into disjoint cells and only the centers of these cells serve as the positions of influence. In this way, the comparison between influence maps is reduced to cell comparisons.

We use *progressive refinement approach* to build approximate influence maps by allowing errors. Assume that the target geographical plane is a square of length L. Given a resolution R, we divide the plane into $\frac{L}{R} \times \frac{L}{R}$ cells. For each cell, we use the center of the cell to approximate the influences exerted on this cell by other objects in the neighbouring cells. The parameter R determine the resolution of this approximation. As long as R is sufficiently small, our model will provide a good approximation. We denote this the approximation of S to be $\widehat{S}$. One issue is to estimate the upper of this approximation error. We define the **Influence Error (IErr)** as follows.

Definition 7 *Suppose we represent the approximate influence map $\widehat{S}$ as a $[n \times m]$ matrix. For any granularity coefficient, c, the refined approximate influence map $\widehat{S'}$ is a $[c \cdot n \times c \cdot m]$ matrix. The difference in the two influence maps is given by:*

$$IErr(\widehat{S'}, \widehat{S}) = \frac{1}{|\widehat{S'}|} \times \sum_{every\ cell \in \widehat{S'}} \frac{\parallel cell_{\widehat{S'}} - cell_{\widehat{S}} \parallel}{MAX(cell_{\widehat{S'}}, cell_{\widehat{S}})} \tag{5}$$

where $|\widehat{S'}|$ denotes the matrix size of $\widehat{S'}$, $cell_{\widehat{S'}}$ is a single cell at $\widehat{S'}$, and $cell_{\widehat{S}}$ is the cell which covers $cell_{\widehat{S'}}$ at matrix $\widehat{S}$.



(a) Influence Maps (b) Intersection

Figure 5. An example to compute influence error

For example, Figure 5 shows $\widehat{S}$ and $\widehat{S}'$, respectively, and one view of the intersection plane. In this example, $IErr$ is the volume of the shaded areas in the right part of Figure 5. $IErr = \frac{1}{4} \times (\frac{|60-50|}{60} + \frac{|65-50|}{65} + \frac{|45-50|}{50} + \frac{|40-50|}{50}) \approx \frac{1}{4} \times 0.69 = 0.17$.

Note that although the term "resolution" used here is similar to the bin width in histogram theory in statistics [10], our approximation approach is different. In statistics, a fine bin width is selected to avoid under-smoothing while estimating data distribution [11], whereas the goal in this paper is to compare multiple influence maps (estimators in KDE) efficiently within an acceptable error tolerance. In this sense, bin width selection in histogram theory is not suitable for our mining requirement. Hence, we design a new mechanism to select the proper resolution for a given error bound.

Lemma 5 *As* $\theta = \frac{R}{\sigma}$, *the error bound IErr between the approximate influence map of resolution* r, $\widehat{S|_{R=r}}$, *and the space of resolution* $\frac{r}{2}$, $\widehat{S|_{R=r/2}}$, *is*

$$IErr(\widehat{S|_{R=r}}, \widehat{S|_{R=r/2}}) \leq \frac{1 - e^{-\frac{R^2}{16\sigma^2}} e^{-\frac{kR}{4\sigma}} + e^{\frac{kR}{4\sigma}} - e^{\frac{R^2}{16\sigma^2}}}{2}, \tag{6}$$

where $0 \leq k \leq 3$.

PROOF: Appendix A gives the proof for Formula 12. Here, we extend the proof to Formula 6. This is done in two steps. First, for one particular cell and a set of objects, Formula 12 holds because the influence from a set of n objects to the center position p of one cell is $\frac{1}{n}\sum_{i=1}^{n} inf(o_i, p)$ given by Definition 3. Next, for all cells on the plane and the set of objects, the influence error is essentially the normalized combination of every singular cell given by Definition 7, so Formula 12 still holds. Hence, we conclude $\widehat{S(\cdot)|_{R=r}} = \sum_{i=1}^{n} \sum_{all\ cells} inf(o_i, cell\ center)$. □

Figure 6 shows the error bounds as we vary k, where the x- coordinate is the ratio $\theta = \frac{R}{\sigma}$, y- is the possible fluctuation over the real influence. The $k = 3$ curve is the *error bound* when $0 \leq k \leq 3$. The worst case error occurs when a split is performed on the marginal cells of an influence unit. In practice, this does not happen often. As a result, the influence fluctuation is far less than the error bound. The real error curve will depends on the data distribution, which is supported by experiments in Section 5.1. In fact, we have the Theorem 1 no matter the data distributions are.

Theorem 1 *As* $\theta = \frac{R}{\sigma} \to 0$, *the approximate influence map* $\widehat{S}$ *converges to the real influence map* S.

PROOF: With an initial resolution r, we obtain the approximation space $\widehat{S|_{R=r}}$. Next, we halve the resolution to obtain its finer approximation space $\widehat{S|_{R=r/2}}$. Lemma 5 gives the upper bound and lower bound of the ratio

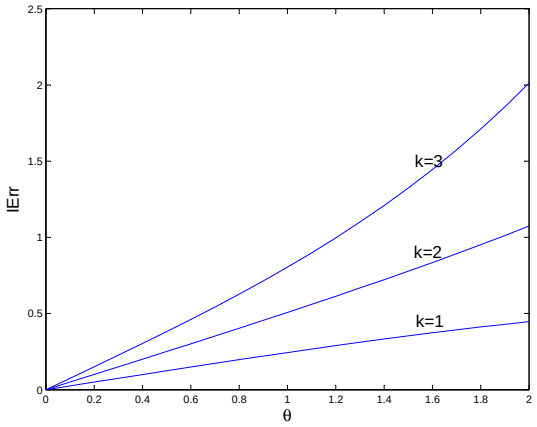


Figure 6. The error bound of influence error

Algorithm 1 BuildApproSpace

Input: Dataset O, kernel deviation σ, inital resolution r and error bound min_err.

Output: Approximate influence map $\widehat{S(\cdot)}$.

1: Initialize two approximation spaces $S_b = \widehat{S(\cdot)|_r}$ and $S_f = \widehat{S(\cdot)|_{r/2}}$;
2: **while** $(IErr(S_f, S_b) > min\_err)$ **do**
3: $S_b \leftarrow S_f$;
4: $r \leftarrow \frac{r}{2}$;
5: Initialize $S_f = \widehat{S(\cdot)|_{r/2}}$;
6: **end while**
7: return S_f;

of the two approximation spaces. By iterating this operation k steps, we have the approximation spaces of $\theta = \frac{R}{\sigma} = \frac{r}{2^k \times \sigma} \to 0$. From Formula 6 and Figure 6, we have $\frac{\hat{S}}{S} \to 1$, which completes the proof. □

Algorithm. Let $O = \{o_1, o_2, \ldots, o_n\}$ be a set of objects of one particular feature. To obtain the approximate influence map of this feature, we employ the BuildApproSpace algorithm.

Line 1 of Algorithm 1 builds two spaces, S_f and S_b. S_f is implemented as a $\frac{L}{r} \times \frac{L}{r}$ matrix while S_b is a $\frac{2L}{r} \times \frac{2L}{r}$ matrix, where L is the plane width. For each object $o_i \in O$, we superimpose a minimal bounding rectangle (MBR) of side 6σ onto the two spaces, centering at the position of o_i. This results in the updates of element values on the two spaces respectively.

To compare the two matrices S_f and S_b, for each element of S_b, we find the corresponding four elements in matrix S_f and obtain the absolute difference among them. Line 2 computes the approximate error within each cell of S_b individually, and take the arithmetic average which is given by Definition 7.

If the approximation error is greater than the user specified parameter min_err, we initialize a new matrix at half the resolution. We compute the approximation error of this finer resolution space. This process repeats until the error is less than min_err, and S_f is the final approximate influence map, which is guaranteed by Theorem 1.

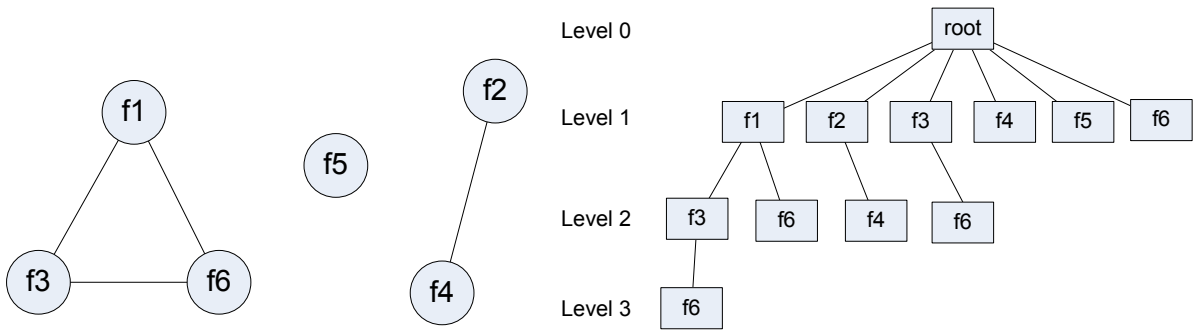
4.2 Pattern Growth and Pruning

Lemma 4 indicates that the interaction measure satisfies the downward closure property. In other words, a candidate pattern is possible to be valid only if all of its subpatterns are valid. This allows many SIPs to be pruned during the mining process.

However, the number of valid SIPs can be very large, since a valid SIP of n features has $(2^n - 1)$ subsets that are also valid SIPs. The majority of these valid SIPs are redundant as their interaction can be inferred from their superset. Avoiding the generation of these redundant SIPs can significantly improve mining efficiency and save memory space. Current state-of-the-art maximal pattern mining algorithms [3, 9] use a search tree structure to facilitate depth-first search to find frequent itemsets, but they cannot deal with maximal interaction pattern mining problem directly.

Motivated by the idea of maximal pattern mining algorithms, we employ a depth-first search with "look ahead". A tree structure called *interaction tree* is used to facilitate the mining process. It is similar to the search tree in [3, 9], but with one important extension. Each node at level 2 denotes an interaction pattern of 2 features, and its associated influence map. Algorithm 2 gives the details.

Assuming that we have obtained all the valid interaction patterns of size 2, denote by C_2. In order to visualize the relationships among feature sets, an *interaction graph* can be constructed beforehand, in which each feature is a node in the graph, and two nodes are connected by one edge if they are correlated (or exist in C_2). Obviously, the interaction graph is an undirected graph due to the symmetric property given in Lemma 3. For example, the correlated pairs from the database forms an interaction graph in Figure 7(a). There are four edges indicating four pairs of correlated features.



(a) An interaction graph (b) An interaction tree

Figure 7. Data Structure for Mining Maximal SIPs

We further transform the interaction graph to an interaction tree as follows. A root node is first created at level 0. At level 1, we create a node for each feature as a child of the root, and the order of children follows sthe lexigraphical order. In the subsequent levels, for each node u at level k ($k > 1$) and for each right sibling v of u, if $((u, v))$ is connected in the interaction graph (namely $\{u, v\}$ is an interaction pattern), we create a child node for u with the same label of v. For each node, we could enumerate one candidate pattern, with prefix feature set of its parent node by concatenating the feature in this node. For example, we construct the tree shown in Figure 7(b) based on Figure 7(a).

Note that if a pattern p is not a valid SIP, then any longer pattern that contains p cannot be a valid SIP. This allows us to effectively prune off unnecessary computations. Further, the structure of the interaction tree always forces the evaluation of the longest patterns first. This implies that if the longest pattern is a valid SIP, then we do not need to evaluate any of its sub-patterns.

4.3 Interaction Tree Traversal

The evaluation of SIP is performed by Algorithm 2. Given a feature node f_n in the interaction tree, Line 1 obtains the parent node of f_n. Line 2 forms a candidate pattern P_{cand} by backtracking from the current node to root of interaction tree. As an example, for the feature node f_6 at level 3 in Figure 7(b), we can backtrack f_6 to form a candidate pattern $\{f_1, f_3, f_6\}$ with prefix $\{f_1, f_3\}$.

Algorithm 2 $EvalSIP(f_n, min\_I, C)$

```
1: parNode ← the parent node of f_n;
2: backtracking fn till root to form a candidate P_cand;
3: if P_cand has a superpattern in C then
4:    set f_n to be a delay node;
5:    EvalSIP(fn's child node, min_I);
6: else
7:    parS = a unit matrix;
8:    while parNode is a delay node do
9:       parS = Interaction(parS, S(parNode));
10:      parNode ← the parent node of parNode;
11:   end while
12:   parS2 ← the influence map of parNode;
13:   parS = Interaction(parS, parS2);
14:   fS = Interaction(parS, S(f_n));
15:   if fS > min_I then
16:      add P_cand to C;
17:      call EvalSIP(f_n's child node, min_I, C);
18:   else
19:      call EvalSIP(f_n's sibling node, min_I, C);
20:   end if
21: end if
```

Since the computation of influence map interaction is expensive, we postpone this computation until it becomes necessary. As long as there is one superset of P_{cand} in C, this computation can be delayed. Line 4 sets this node to be a *delay* node and Line 5 propagates this to its children nodes. Lines 7-13 recover the influence map of the prefix. For example, suppose we already have a maximal SIP $\{f_1, f_2, f_4, f_5\}$ in maximal SIP set C. The candidate patterns $\{f_1, f_4\}$ and $\{f_1, f_4, f_5\}$ can be exempt from evaluating as they are both subpatterns of $\{f_1, f_2, f_4, f_5\} \in C$. Here f_4 and f_5 are marked *delay* nodes in interaction trees. If we need to evaluate another candidate pattern $\{f_1, f_4, f_5, f_6\}$, the interaction computation of its prefixes $\{f_1, f_4\}$ and $\{f_1, f_4, f_5\}$ becomes necessary. So the influence map computation will start from f_4 via f_5 along the path to f_6. The pseudocode of this operation is given in Lines 7-13. Finally, Line 13 obtains the final influence map of prefix, $parS$.

Lines 14-19 compute the interaction between the influence maps of the prefix node and the current node in a depth-first manner. If the interaction is no less than the threshold min_I, this candidate pattern is valid, and it will be added to the maximal pattern set C in Line 16.

Although algorithm $EvalSIP(\cdot)$ is devised to find maximal SIPs, it is capable to discover all valid SIPs. The idea is straightforward: the subpatterns checking in maximal SIP set C is skipped, which is achieved by deleting Lines 3-11 in Algorithm 2.

4.4 Algorithm PROBER

We now present the Algorithm PROBER to mine spatial interaction patterns. The algorithm incorporates the pattern enumeration technique into the mining process. Algorithm PROBER takes as input the spatial database D, the influence error threshold min_err, the interaction measure threshold min_I, and outputs the set of maximal SIPs.

Algorithm 3 PROBER

Input: D: the spatial database;
σ: kernel deviation;
min_err: influence error threshold;
min_I: interaction threshold.

Output: C: the set of SIPs.

1: Let RF to be all features in D;
2: /*Phase 1: impose approximate influence map*/
3: **for** each feature $f_i \in RF$ **do**
4: call $BuildApproSpace(\cdot)$ to build the influence map;
5: **end for**
6: /*Phase 2: build interaction tree*/
7: $C = \emptyset$;
8: Impose an ordering on RF;
9: **for** each feature pair (f_i, f_j), where $f_i \prec f_j$ **do**
10: evaluate feature pair (f_i, f_j);
11: if $I(f_i, f_j) \geq min\_I$, add pattern $\{f_i, f_j\}$ to C;
12: **end for**
13: build the interaction tree T_{col} based on C;
14: /*Phase 3: mine maximal SIP*/
15: call $EvalSIP(T_{col}.root, min\_I, C)$;
16: return C;

Line 1 finds the feature set from the dataset. Lines 2-4 build the approximate influence maps for each feature, by calling the $BuildApproSpace(\cdot)$ algorithm. To facilitate the next mining phase, the approximate influence maps of all features are required to be superimposed using the same resolution. Therefore, the halt condition in $BuildApproSpace(\cdot)$ is modified to be "If the maximal $IErr(S_b, S_f)$ of all features is greater than min_err, then do the next iteration".

Lines 6-9 discover the interaction in all feature pairs combination. In particular, Line 7 computes the interaction of the one feature pair by taking the minimal value between each element pair and summing up all the minimal values. If the measure is greater than min_I, this feature pair is considered to be correlated.

Line 11 builds the interaction tree using the set of interaction pairs as the tree proposed in [5]. Line 12 invokes algorithm $EvalSIP(\cdot)$ to recursively visit the necessary feature nodes in interaction tree, starting from the root node of tree. Finally, Line 13 returns the maximal pattern set C.

Continuing with our example in Figure 1, we assume σ=10 and min_I= 0.3, Table 1 shows the mining process of PROBER. The mining stops at level 2 because the pattern $\{f_1, f_3\}$ is not a valid SIP, hence the pattern $\{f_1, f_2, f_3\}$ is pruned. As a result, the maximal SIPs are $\{\{f_1, f_2\}$ and $\{f_2, f_3\}\}$.

Complexity Analysis. Let θ=σ/R where R is final resolution after multiple iterations. To build the influence map (i.e. $BuildApproSpace(\cdot)$ algorithm), an influence unit of range $6\sigma \times 6\sigma$ is computed for each instance in the database, thus it needs $(6\sigma/R)^2$=$(6/\theta)^2$ distance computation. The overall computational complexity to build the influence maps is $O(n(6/\theta)^2)$, where n is the database size. Since there is one influence map matrix for each feature, the space complexity is $O(f(\frac{L}{R})^2)$ where L is the plane length, f is the feature number.

The PROBER algorithm, in the worst case, could generate $\binom{f}{\lfloor \frac{f}{2} \rfloor}$ maximal SIPs, and each of them requires $\lfloor \frac{f}{2} \rfloor$ influence map comparison along the path from root to the leaf node of the maximal SIP.

Table 1. Mining SIPs by influence model

level=1		
f_1	$S(f_1)$ (see Figure 4(a))	$I(f_1)$=1
f_2	$S(f_2)$ (see Figure 4(b))	$I(f_2)$=1
f_3	$S(f_3)$ (see Figure 4(c))	$I(f_3)$=1
level=2		
$\{f_1, f_2\}$	$S(f_1, f_2)$ (see Figure 4(d))	$I(f_1, f_2)$=0.58
$\{f_1, f_3\}$	$S(f_1, f_3)$	$I(f_1, f_3)$=0.27
$\{f_2, f_3\}$	$S(f_2, f_3)$ (see Figure 4(e))	$I(f_2, f_3)$=0.38

Each influence map comparison requires the computation of complexity $O((\frac{L}{R})^2)$. Hence, the overall computational complexity in mining phase, is $O(\binom{f}{\lfloor \frac{f}{2} \rfloor} \times \lfloor \frac{f}{2} \rfloor \times (\frac{L}{R})^2)$. The space complexity include the space to store interaction tree and influence matrixes. The interaction tree has maximal 2^f nodes, and each node store an influence matrix. As the space required for influence matrix dominates, the worst space requirement is $O((2^f - f) \times (\frac{L}{R})^2)$.

In summary, the computational complexity of PROBER is $O(n(6/\theta)^2 + \binom{f}{\lfloor \frac{f}{2} \rfloor} \times \lfloor \frac{f}{2} \rfloor \times (\frac{L}{R})^2)$, and the space complexity of PROBER is $O(2^f (\frac{L}{R})^2)$.

5 Experiment Study

In this section, we present the results of experiments to evaluate the performance of PROBER. We also compare PROBER with existing collocation algorithms FastMiner [15] and TPMiner [12]. In order to set reasonable comparison, we generate two versions of PROBER, PROBER_ALL to discover all patterns, and PROBER_MAX to discover only maximal patterns. Note that while comparing the effectiveness of PROBER with FastMiner and TPMiner may not be appropriate due to the different interesting measures used, however, we could treat FastMiner and TPMiner as good baselines w.r.t both effectiveness and scalability issues. Table 2 show the parameter counterparts between influence model and distance model. In the following experiments, we assign the identical values to the parameter counterparts, e.g. $\sigma = d = 50$ and $min\_I = min\_prev = 0.4$.

Table 2. Parameter counterparts

Distance Model	Influence Model
distance threshold: d	influence deviation: σ
minimal prevalence: min_prev	minimal interaction: min_I

Synthetic Datasets: We extend the synthetic data generator in [12] to generate the synthetic spatial databases with Gaussian noise. All the data are distributed on the plane of 8192×8192. The synthetic datasets are named using the format "Data-(m)-(n)-(d)-(N)" where m is the confidence feature number, n is the non-confidence feature number, d is the distance threshold and N is the number of instances in the dataset. For example, DATA-8-2-50-200k is a dataset which contains 8 confident features, 2 noise features, distance threshold to be 50, and a total of 200k instances. For each object, we assign a Gaussian noise of 0 mean and σ_e, say 5, deviation on each dimension.

Real-life Datasets: The real-life dataset used in our experiments is the *DCW environmental data.* We downloaded 8 layers of Minnesota state from Digital Chart of the World [3] (DCW). Each layer is regarded as a feature in our experiments as shown in the left of Table 5. We further map the dataset on a formal 8192×8192 2D plane.

All the algorithms are implemented in C++. The experiments were carried out on a Pentium 4 3Ghz PC with 1GB of memory, running Windows XP.

5.1 Performance of Influence Map Approximation

In this section, we evaluate the convergence and efficiency of the *BuildApproSpace* algorithm on both synthetic data Data-6-2-100-50k and DCW data. We assign the initial resolution $r = 256$ and $min\_err = 0.05$, as well as $\sigma = 100$ for Data-6-2-100-50k and $\sigma = 50$ for DCW data. Table 3 and 4 shows the results. In both tables, each row gives the influence error $IErr$ for each iteration. We can see that the $IErr$ converges to zero with each iteration.

Table 3. Convergence on DCW data

Iteration	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	MAX
1	0.16	0.02	0.15	0.13	0.16	0.04	0.17	0.17	0.17
2	0.10	0.01	0.09	0.08	0.10	0.03	0.11	0.11	0.11
3	0.06	0.01	0.05	0.05	0.06	0.02	0.06	0.07	0.07
4	0.03	0.00	0.03	0.03	0.03	0.01	0.04	0.04	0.04

Table 4. Convergence on Data-6-2-100-50k

Iteration	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	MAX
1	0.17	0.17	0.17	0.17	0.17	0.17	0.25	0.24	0.25
2	0.10	0.10	0.10	0.10	0.10	0.10	0.15	0.14	0.15
3	0.05	0.06	0.06	0.06	0.06	0.06	0.08	0.08	0.08
4	0.03	0.03	0.03	0.03	0.03	0.03	0.04	0.04	0.04

The time taken for each iteration is shown in Figure 8. We observe that the runtime increases quadratically as the number of iteration increases, as the resolution and θ are decreased by half. This is expected because a finer resolution and smaller θ will cause a quadratic increase in both time and space complexity. On the other hand, the runtime is linear to the database size for each iteration, as the DCW data contains 2837 objects as a whole and the synthetic dataset contains 50k objects. The results in Figure 8 are consistent with the analysis in Section 4.4.

5.2 Effectiveness Study

In this set of experiments, we show that PROBER_ALL algorithm is more robust than FastMiner and TopologyMiner as the variation of deviation/distance threshold. As the results of FastMiner and TopologyMiner are exactly same, we only compare PROBER_ALL with TopologyMiner in our experiment.

[3] http://www.maproom.psu.edu/dcw

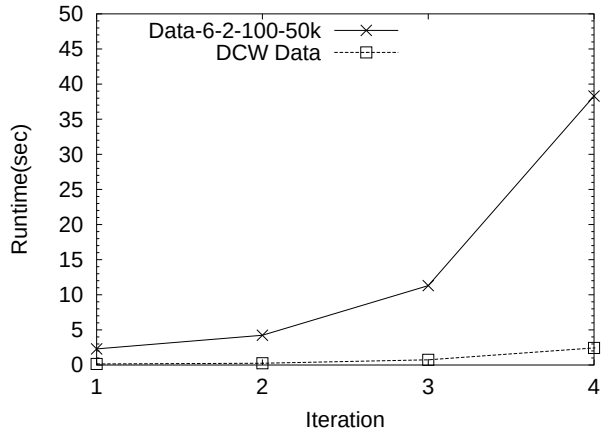


Figure 8. Convergence Performance

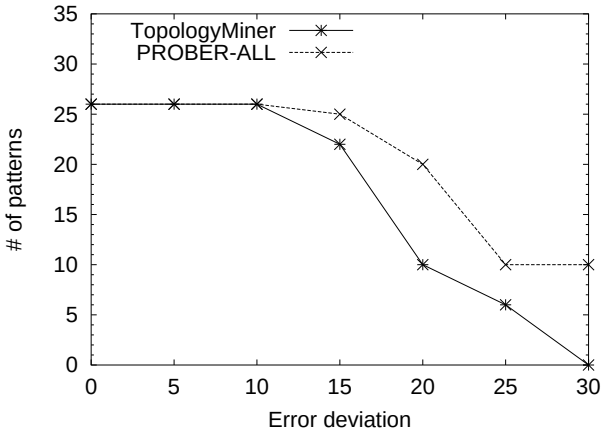
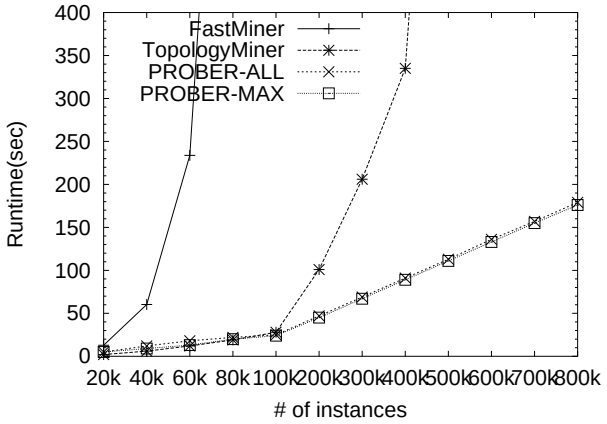
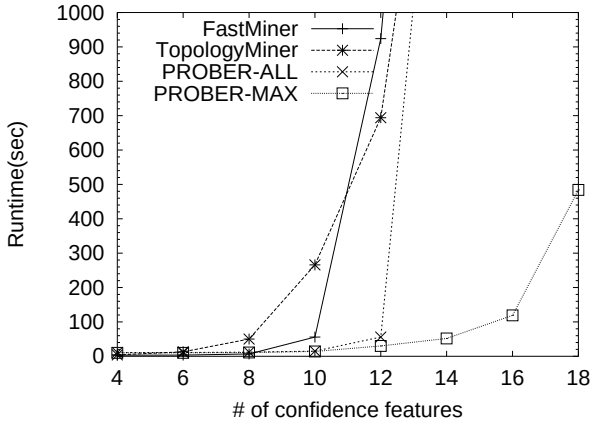


Figure 9. Effectiveness study

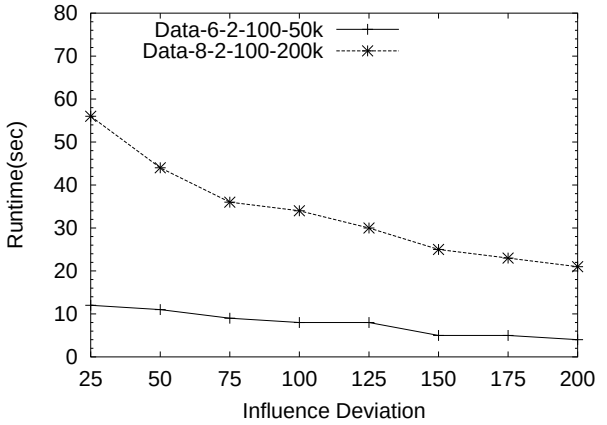


(a) Effect of DB size

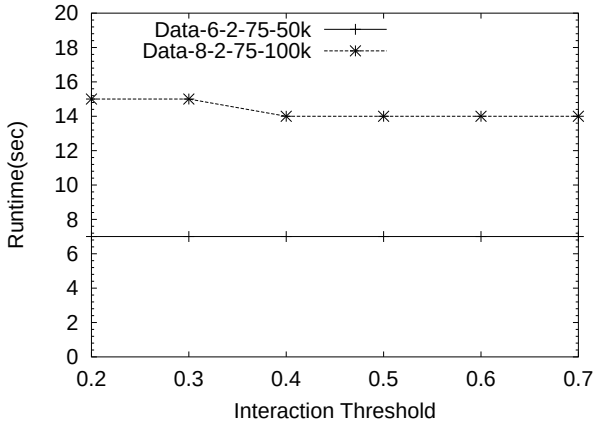


(b) Effect of confidence feature number

Figure 10. Scalability study



(a) Effect of influence deviation



(b) Effect of interaction

Figure 11. Sensitivity study

Table 5. Feature Description and Patterns Comparison of DCW Dataset

FID	Name	# of Points
f_0	Populated Place	517
f_1	Drainage	6
f_2	Drainage Supplemental	1338
f_3	Hypsography	72
f_4	Hypsography Supplemental	687
f_5	Land Cover	28
f_6	Aeronautical	86
f_7	Cultural Landmarks	103

d/σ	Distance Model	Influence Model
50	NA	$\{f_0, f_2\}$, $\{f_2, f_4\}$
100	$\{f_0, f_2\}$, $\{f_2, f_4\}$	$\{f_0, f_2\}$, $\{f_0, f_4\}$, $\{f_0, f_6\}$, $\{f_2, f_4\}$
150	$\{f_0, f_2\}$, $\{f_0, f_4\}$, $\{f_2, f_4\}$	$\{f_0, f_2\}$, $\{f_0, f_3\}$,$\{f_0, f_4\}$, $\{f_0, f_6\}$,$\{f_0, f_7\}$, $\{f_2, f_3\}$, $\{f_2, f_4\}$, $\{f_2, f_6\}$, $\{f_3, f_4\}$,$\{f_3, f_7\}$, $\{f_4, f_6\}$,$\{f_4, f_7\}$, $\{f_6, f_7\}$
200	$\{f_0, f_2, f_4\}$, $\{f_3, f_4\}$	$\{f_0, f_2, f_4\}$, $\{f_0, f_2, f_6\}$, $\{f_0, f_6, f_7\}$, $\{f_0, f_3\}$, $\{f_2, f_3\}$, $\{f_2, f_7\}$, $\{f_3, f_4\}$,$\{f_3, f_6\}$,$\{f_3, f_7\}$, $\{f_4, f_6\}$,$\{f_4, f_7\}$

We first use a synthetic dataset Data-5-0-50-5k to evaluate effectiveness. Without noise, the expected patterns will be the maximal pattern of 5 confident features and all its sub-patterns, i.e. 26 patterns (2^5-5-1=26). We integrate different Gaussian noise of identical mean 0 but different deviation ranging from 1 to 30. The comparison measure is the number of interaction patterns discovered, including maximal and non-maximal ones. The results are shown in Figure 9. From this figure, we observe that both PROBER_ALL and TopologyMiner can find the whole set of possible patterns while error deviation is less than 10. This is expected because the small error deviations do not have impact on the influence deviation. Therefore TopologyMiner can find all of the patterns with noise of deviation 10. On the other hand, the patterns found by TopologyMiner show greater decrease than the ones by PROBER_ALL, as the increase of error deviation.

Next, we apply PROBER_ALL on the DCW environment data. We set the interaction threshold min_I to be 0.4. The mining results are shown in the right of Table 5. We observe that regardless of how the σ varies, the patterns discovered by PROBER_ALL are always a superset of those found by the two distance model-based techniques, FastMiner and TopologyMiner. In particular, when $\sigma = 200$, we find that {populated place, drainage supplemental, hypsography supplemental} and {populated place, aeronautical, cultural landmarks} are missed by the distance model-based techniques but are discovered as SIPs.

5.3 Scalability

In this set of experiments, we demonstrate the scalability of both PROBER_ALL and PROBER_MAX. We set the number of features to 10 (including non-noise and noise features), and generate twelve datasets Data-8-2-50-{20k, 40k, 60k, 80k, 100k, 200k, ..., 800k}. We compare the performance of PROBER with FastMiner and TopologyMiner by varying the total number of instances. Figure 10(a) shows that

both FastMiner and TopologyMiner increase exponentially as the number of instances increases while PROBER shows a linear increase. This is expected because the time complexity for the distance model is polynomial time of the number of instances while PROBER_ALL and PROBER_MAX are linear to the number of instances during the database scan and is independent of the database size during mining phase. In further observation, PROBER_MAX is slightly faster than PROBER_ALL because PROBER_MAX only detects the maximal patterns which can save the mining cost.

We also set the database size at 20k instances and generate eight datasets Data-{4,6,8,10,12,14,16,18}-0-50-20k to evaluate the three algorithms. The results are shown in Figure 10(b). Both FastMiner and TopologyMiner do not scale well w.r.t the number of non-noise features. TopologyMiner allows pattern growth in a depth-first manner, but the extraction of project databases requires much time and space. PROBER_MAX shows the best scalability compared to the other algorithms, although the algorithm slows down when the number of features exceeds 16. This is because of the large confidence features results in the exponential growth of its interaction tree.

5.4 Sensitivity

Finally, we examine the effect of two parameters, influence deviation σ and interaction threshold min_I, on the performance of PROBER. Due to the intrinsic difference between influence model and distance model, it is unfair to compare the sensitivity performance of the two models. Therefore, we only include the influence model in this experiment.

Effect of Influence Deviation (σ). We first evaluate the effect of the influence deviation on PROBER_MAX. The two datasets used in this experiment are Data-6-2-100-50k and Data-8-2-100-200k, which imply that the patterns will be valid once the σ surpass 100. Figures 11(a) gives the results. PROBER_MAX run faster as the increase of σ. This is expected due to two reasons: 1) The time complexity of PROBER are inversely related to the σ, consistent to the complexity analysis in Section 4.4; 2) Bigger σ implies smoother distribution of influence maps, so it incurs less iteration rounds to build approximate spaces with error $IErr$, which saves the cost. On the contrary, the algorithms of distance model are sensitive to distance threshold due to the tremendous increase of time cost. Please refer to [12] for details.

Effect of Interaction Threshold (min_I). We evaluate the three algorithms on two dataset Data-3-3-50-20k and Data-5-5-50-50k with potential prevalence is 0.5, which implies that the PROBER may find many patterns while $min\_I < 0.5$. From Figures 11(b), PROBER_MAX are not sensitive to min_I because the mining cost is not the dominant factor compared with the cost to build approximate spaces. On the contrary, the runtime of FastMiner and TopologyMiner will decrease as prevalence increases. This is expected because less patterns become frequent as the increase of prevalence, which leads to reduced mining cost for the two algorithms.

6 Conclusion

In this paper, we have introduced an influence model to discover valid Spatial Interaction Patterns (SIPs) from spatial databases. Compared to the distance model used in existing works, the influence model consider the spatial affinity in terms of continuous functions instead of discrete functions. This leads to more meaningful mining results. Another advantage of the influence model is its ability to avoid expensive join operations that are traditionally required to discover the relationship among spatial instances. We also analyzed the bounds for computational error and designed an approximate mining algorithm PROBER to find maximal SIPs. The experiment results on both synthetic and real-life datasets demonstrated that PROBER is effective and scalable.

References

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *VLDB '94*, pages 487–499. Morgan Kaufmann, 12–15 1994.

[2] P. M. Dixon. Ripley's k function. *Encyclopedia of Environmetrics*, 3:1796– 1803, 2002.

[3] K. Gouda and M. J. Zaki. Efficiently mining maximal frequent itemsets. In *ICDM*, pages 163–170, 2001.

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

[5] Y. Ke, J. Cheng, and W. Ng. Mining quantitative correlated patterns using an information-theoretic approach. In *KDD '06*, pages 227–236, New York, NY, USA, 2006. ACM Press.

[6] K. Koperski and J. Han. Discovery of spatial association rules in geographic information databases. In M. J. Egenhofer and J. R. Herring, editors, *SSD '95*, volume 951, pages 47–66. Springer-Verlag, 6–9 1995.

[7] Y. Morimoto. Mining frequent neighboring class sets in spatial databases. In *KDD*, pages 353–358, 2001.

[8] B. D. Ripley. *Spatial Statistics*. Wiley, 1981.

[9] J. Roberto J. Bayardo. Efficiently mining long patterns from databases. In *SIGMOD '98: Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, pages 85–93, New York, NY, USA, 1998. ACM Press.

[10] D. W. Scott. *Multivariate Density Estimation: Theory, Practice, and Visualization*. Wiley, 1992.

[11] J. S. Simonoff. *Smoothing Methods in Statistics*. Springer, 1996.

[12] J. Wang, W. Hsu, and M. L. Lee. A framework for mining topological patterns in spatio-temporal databases. In *CIKM '05*, pages 429–436, New York, NY, USA, 2005. ACM Press.

[13] J. S. Yoo, S. Shekhar, and M. Celik. A join-less approach for co-location pattern mining: A summary of results. In *ICDM*, pages 813–816, 2005.

[14] J. S. Yoo, S. Shekhar, J. Smith, and J. P. Kumquat. A partial join approach for mining co-location patterns. In *GIS '04*, pages 241–249, New York, NY, USA, 2004.

[15] X. Zhang, N. Mamoulis, D. W. Cheung, and Y. Shou. Fast mining of spatial collocations. In *KDD '04*, pages 384–393, New York, NY, USA, 2004. ACM Press.

A Influence Approximation

Our idea to determine the appropriate resolution is as follows. First, we partition the plane into a coarse granularity. Then we recursively perform a split operation to divide each cell into 4 sub-cells. These sub-division steps will assign a finer granularity which is exactly half of the previous resolution. In this way, we can compute the effect of finer resolution, and eventually arrive at the appropriate resolution.

Figure 12 shows the splitting strategy, where o is the position of object and p is the center of a big grid of side R, the distance from o to p is indicated by symbol d. After unform splitting, this big grid is partitioned into four subgrids, each of which has a side $R/2$. The distances from o to the center of each subgrid are d'_1, d'_2,d'_3,d'_4 respectively.

In the following analysis of the bounds of the approximation, we only consider the case where the objects are distributed in the east quarter area. Without loss of generality, any other distributions can be transformed into this case by rotating the cell.

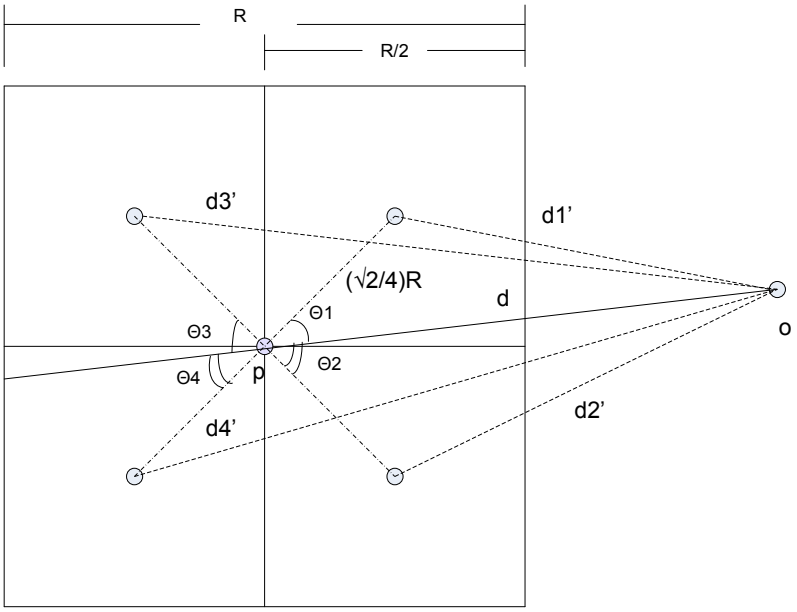


Figure 12. Cell splitting case

After splitting, we will have the following equations according to the *Cosine Theorem*:

$$\begin{cases} d_1'^2 = d^2 + \frac{1}{8}R^2 - \frac{\sqrt{2}}{2}dR\cos\theta_1 \\ d_2'^2 = d^2 + \frac{1}{8}R^2 - \frac{\sqrt{2}}{2}dR\cos\theta_2 \\ d_3'^2 = d^2 + \frac{1}{8}R^2 + \frac{\sqrt{2}}{2}dR\cos\theta_3 \\ d_4'^2 = d^2 + \frac{1}{8}R^2 + \frac{\sqrt{2}}{2}dR\cos\theta_4 \end{cases}$$

Since $\theta_1 + \theta_2 = \pi/2$ and $\theta_3 + \theta_4 = \pi/2$, we have the following two constraints:

$$\begin{cases} 1 < \cos\theta_1 + \cos\theta_2 < \sqrt{2} \\ 1 < \cos\theta_3 + \cos\theta_4 < \sqrt{2} \end{cases}, \text{ and}$$

$0 < \cos\theta_1, \cos\theta_2, \cos\theta_3, \cos\theta_4 < 1$

The summation of the first two influence units is

$$Inf_1 + Inf_2 = \frac{R^2}{4} \cdot (e^{-\frac{d_1'^2}{2\sigma^2}} + e^{-\frac{d_2'^2}{2\sigma^2}})$$

$$= \frac{R^2}{4} \cdot e^{-\frac{d^2}{2\sigma^2}} \cdot e^{-\frac{R^2}{16\sigma^2}} \cdot (e^{\frac{\sqrt{2}dR\cos\theta_1}{4\sigma^2}} + e^{\frac{\sqrt{2}dR\cos\theta_2}{4\sigma^2}}) \quad (7)$$

Let $f(d) = e^{\frac{\sqrt{2}dR\cos\theta_1}{4\sigma^2}} + e^{\frac{\sqrt{2}dR\cos\theta_2}{4\sigma^2}} = e^{\frac{\sqrt{2}dR\cos\theta_1}{4\sigma^2}} + e^{\frac{\sqrt{2}dR\sin\theta_1}{4\sigma^2}}$, with $\theta_2 = \pi/2 - \theta_1$. $f(d)$ reaches a local

minimal at $\theta_1 = 0$ and a local maximal at θ_1=$\pi/4$. So we have

$$2 < 1 + e^{\frac{\sqrt{2}dR}{4\sigma^2}} \leq f(d) \leq 2e^{\frac{dR}{4\sigma^2}}. \tag{8}$$

From Formula (7) and (8), we have

$\frac{R^2}{2} \cdot e^{-\frac{d^2}{2\sigma^2}} \cdot e^{-\frac{R^2}{16\sigma^2}} < Inf_1 + Inf_2 \leq \frac{R^2}{2} \cdot e^{-\frac{d^2}{2\sigma^2}} \cdot e^{-\frac{R^2}{16\sigma^2}} \cdot e^{\frac{dR}{4\sigma^2}}$

So the influence error at the first two sub-cells is

$IErr_{1,2}(\cdot) = \frac{\|(Inf_1+Inf_2)-Inf/2\|}{Inf_1+Inf_2}$

$$\leq \frac{\frac{R^2}{2} \cdot e^{-\frac{d^2}{2\sigma^2}} \cdot (e^{-\frac{R^2}{16\sigma^2}} \cdot e^{\frac{dR}{4\sigma^2}} - 1)}{\frac{R^2}{2} \cdot e^{-\frac{d^2}{2\sigma^2}} \cdot e^{-\frac{R^2}{16\sigma^2}}} = e^{\frac{dR}{4\sigma^2}} - e^{\frac{R^2}{16\sigma^2}} \tag{9}$$

Similarly, the summation of the last two influence units is

$\frac{R^2}{2} \cdot e^{-\frac{d^2}{2\sigma^2}} \cdot e^{-\frac{R^2}{16\sigma^2}} \cdot e^{-\frac{dR}{4\sigma^2}} \leq Inf_3 + Inf_4 < \frac{R^2}{2} \cdot e^{-\frac{d^2}{2\sigma^2}} \cdot e^{-\frac{R^2}{16\sigma^2}}$
So the influence error at the last two sub-cells are

$$IErr_{3,4}(\cdot) = \frac{\| \, Inf/2 - (Inf_3 + Inf_4) \, \|}{Inf/2} \leq 1 - e^{-\frac{R^2}{16\sigma^2}} e^{-\frac{dR}{4\sigma^2}} \tag{10}$$

Combining Formula (9) and (10), we have the influence error

$$IErr(\cdot) = \frac{IErr_{1,2} + IErr_{3,4}}{2} \leq \frac{1 - e^{-\frac{R^2}{16\sigma^2}} e^{-\frac{dR}{4\sigma^2}} + e^{\frac{dR}{4\sigma^2}} - e^{\frac{R^2}{16\sigma^2}}}{2} \tag{11}$$

As $0 \leq d \leq 3\sigma$, we normally substitute $k\sigma$ for d where $0 \leq k \leq 3$. So Formula 11 can be rewrote as

$$IErr(\cdot) \leq \frac{1 - e^{-\frac{R^2}{16\sigma^2}} e^{-\frac{kR}{4\sigma}} + e^{\frac{kR}{4\sigma}} - e^{\frac{R^2}{16\sigma^2}}}{2} \tag{12}$$