A Fast Algorithm for Combinatorial Hotspot Mining Based on Spatial Scan Statistic

Shin-ichi Minato[*] Jun Kawahara[†] Fumio Ishioka[‡] Masahiro Mizuta[§]
Koji Kurihara[¶]

Abstract

It is a popular and classical problem to detect a *hotspot* cluster from a statistical data which is partitioned by geographical regions such as prefectures or cities. *Spatial scan statistic* is a standard measure of likelihood ratio which has been widely used for testing hotspot clusters. In this work, we propose a very fast algorithm to enumerate all combinatorial regions which are more significant than a given threshold value. Our algorithm features the fast exploration by pruning the search space based on the partial monotonicity of the spatial scan statistic. Experimental results for a nation-wide 47 prefectures dataset show that our method generates the highest-ranked hotspot cluster in a time a million or more times faster than the previous naive search method. Our method works practically for a dataset with several hundreds of regions, and it will drastically accelerate hotspot analysis in various fields.

keywords: hotspot detection, data mining, scan statistic, enumeration algorithm, combinatorial problem

1 Introduction

It is a basic and important task to detect where a problem occurs, such as the generation status of infective diseases or hazard maps of natural disasters. *Hotspot detection* is one of the popular and classical problems to detect a cluster region of highly concentrative occurrences in a statistical data which is partitioned by geographical regions such as prefectures or cities. In the literature, there have been many studies on spatial statistics for evaluating hotspot regions, for example, Moran's I statistic [12, 2] based on spatial autocorrelation, Cuzick-Edward's test [4] based on a kind of k-nearest neighbors method, and Tango's index [15] based on a measure of closeness between regions, have been proposed.

The *spatial scan statistic*, proposed by Kulldorff [9], has now widely been used for cluster detection together with an open software tool. This is originally from the one-dimensional scan statistic [13] for scanning temporal data to detect the intervals of frequent occurrences. Spatial scan statistic is the extension of original one to the spatial data, and now it is commonly used for hotspot detection in various fields because of its excellent statistical properties. In scanning spatial data, the number of adjacent regional patterns of hotspot candidates grows exponentially large with respect to the number of regions, so it is too time-consuming to cover all possible candidates unless the number is very small. Most of existing methods put some restrictions in shape or size of the cluster candidates. Kulldorff et al.[10] uses a circular-shaped window to scan the potential cluster region, but a non-circular shaped cluster, such as the shape formed by a river or a road, cannot be detected by that means. To capture arbitrarily shaped clusters, several non-circular scanning techniques have been proposed. *Flexible scan* [16] generates all the adjacent regional patterns up to the size k, and Echelon analysis-based method [11, 7] generates cluster candidates based on Bayesian estimates. Anyway, those methods can detect hotspots only from a restricted set of cluster regions.

In this paper, we consider the combinatorial hotspot detection without any restriction, covering all combinatorial patterns in the power set of the regions, even allowing non-connected patterns. In this problem setting, the number of hotspot candidates becomes further large, but the search space is just a power set of the items without any geographical property, and thus the problem structure becomes simpler rather than original one. We propose a hotspot mining algorithm, named *power set scan*, to enumerate all combinatorial regions which are more significant than a given threshold of the statistic value. Our algorithm features the fast exploration by pruning the search space based on the partial monotonicity of the spatial scan statistic. Experimental results for a nation-wide statistical dataset show that our method generates the highest-ranked hotspot cluster in

[*]Kyoto University.
[†]Nara Institute of Science and Technology.
[‡]Okayama University.
[§]Hokkaido University.
[¶]Okayama University.

Table 1: An example of input data

Region-ID	Population	Occurrence
1 (Hokkaido)	5,401,210	1,004
2 (Aomori)	1,338,465	279
3 (Iwate)	1,289,470	322
4 (Miyagi)	2,324,466	443
$\cdots$	$\cdots$	$\cdots$
47 (Okinawa)	1,461,231	258

a time a million or more times faster than the previous naive search method. Our method works practically for a dataset with several hundreds of regions, and it will drastically accelerate hotspot analysis in various fields.

In the rest of the paper, we briefly review the hotspot detection and the spatial scan statistic in Section 2, and then formulate our problem in Section 3. We present our algorithm and discuss the correctness in Section 4. Experimental evaluations are shown in Section 5, followed by summary and future work.

2 Preliminary – Scan Statistic and Hotspot Detection

2.1 Definitions

In this work, we assume a kind of statistical data which is partitioned by geographical regions such as prefectures or cities. An example of our input data is shown in Table 1. Here, each region has a region-ID and two numerical values: *population* and *occurrence*[1]. Formally, we consider a set of regions $G = \{1, 2, \ldots, m\}$, where $m \in \mathbf{N}$ is the cardinality of G, and then we define the input data $\mathcal{D} = \{(n_1, c_1), (n_2, c_2), \ldots, (n_m, c_m)\}$, where $n_i \in \mathbf{N}$ is the population and $c_i \in \mathbf{N}$ is the occurrence in the region i. We assume $0 \leq c_i \leq n_i$ for any $i \in G$. The *occurrence ratio* in the region i can be written as (c_i/n_i).

For a given set of regions $Z \subseteq G$, let $n(Z)$ and $c(Z)$ denote the population and the occurrence in Z, respectively. Namely,

$$n(Z) = \sum_{i \in Z} n_i, \quad c(Z) = \sum_{i \in Z} c_i \ .$$

Let Z^c denote the complement set of Z, i.e. $Z^c = G \setminus Z$. Clearly $n(Z^c) = n(G) - n(Z)$ and $c(Z^c) = c(G) - c(Z)$.

2.2 Spatial Scan Statistic

Spatial scan statistic is a statistical measure of likelihood ratio for testing whether the event occurrences are fairly incidental or biased by regions, and used for detecting a *hotspot*, a cluster region of significantly concentrative occurrences. This is originally from the one-dimensional scan statistic for scanning temporal data to detect the intervals of frequent occurrences, and now it is commonly used for spatial hotspot detection in various fields because of its excellent statistical properties.

Here we briefly review the idea of the spatial scan statistic [9]. We consider a cluster region Z to be tested, and assume that Z and Z^c may have different occurrence probabilities p_1 and p_2, respectively. We also assume that each event occurs *i.i.d.* in the same cluster region. Then, the null hypothesis leads to $p_1 = p_2$ (non-biased), and the alternative hypothesis leads to $p_1 > p_2$ (biased).

Now we assume the case of Poisson model for event occurrences, then the probability of observing $c(G)$ can be written as:

$$\frac{\exp[-E_G] \cdot (E_G)^{c(G)}}{c(G)!},$$

where $E_G = p_1 n(Z) + p_2 n(Z^c)$ as the expected value of the total occurrence. The density function at the region $i \in G$ is written as:

$$\begin{cases} \dfrac{p_1 \cdot n_i}{E_G} & \text{if } i \in Z, \\ \dfrac{p_2 \cdot n_i}{E_G} & \text{if } i \in Z^c. \end{cases}$$

Thus, the likelihood function for the given observation is expressed as:

$$L(Z, p_1, p_2) = \frac{\exp[-E_G]}{c(G)!} p_1^{c(Z)} p_2^{c(Z^c)} \prod_{i \in G} (n_i)^{c_i} \ .$$

To maximize the likelihood function, we first take the supremum over all p_1 and p_2 for a fixed Z. We then get the estimators $\hat{p_1} = c(Z)/n(Z)$ and $\hat{p_2} = c(Z^c)/n(Z^c)$, and the maximum value of the likelihood function for Z is written as:

$$L(Z) = \frac{\exp[-c(G)]}{c(G)!} \left(\frac{c(Z)}{n(Z)}\right)^{c(Z)} \left(\frac{c(Z^c)}{n(Z^c)}\right)^{c(Z^c)} \prod_{i \in G} (n_i)^{c_i} \ .$$

On the other hand, if we assume the null hypothesis as $p_1 = p_2$, the maximum value of the likelihood function does not depend on Z as:

$$L_0 = \frac{\exp[-c(G)]}{c(G)!} \left(\frac{c(G)}{n(G)}\right)^{c(G)} \prod_{i \in G} (n_i)^{c_i} \ .$$

[1]We may consider not only human health data but also business data, for example, the amount of industrial products and the occurrence of inferior ones.

Let $\lambda(Z)$ denote the ratio of $L(Z)$ and L_0, then we get

$$\lambda(Z) = \begin{cases} \dfrac{\left(\frac{c(Z)}{n(Z)}\right)^{c(Z)} \left(\frac{c(Z^c)}{n(Z^c)}\right)^{c(Z^c)}}{\left(\frac{c(G)}{n(G)}\right)^{c(G)}}, & \text{if } \dfrac{c(Z)}{n(Z)} > \dfrac{c(Z^c)}{n(Z^c)} \\ 1, \quad \text{otherwise.} \end{cases}$$

This is the formula of the spatial scan statistic. Notice that $\lambda(Z)$ has a condition that the occurrence ratio in Z is higher than in Z^c, because the formula is symmetric for exchanging Z and Z^c.

The hotspot cluster Z maximizing $\lambda(Z)$ explains that Z gives the maximum likelihood for observing $c(G)$, provided that the given data can be categorized into the two regional groups Z and Z^c, each of which has a flat occurrence probability under Poisson model. The spatial scan statistic has a good property that we don't have to specify the size of cluster regions beforehand, since the size is automatically decided by maximizing $\lambda(Z)$. Conventionally, we calculate and maximize $\log \lambda(Z)$ instead of $\lambda(Z)$. The computational cost is not so large but the functional behavior analysis is not easy because the formula contains nonlinear operations such as powers or logarithms.

2.3 Hotspot Detection and Testing Procedure

The hotspot detection problem is to find the optimal solution $Z^* \subset G$ which maximizes $\lambda(Z)$. In principle, we may generate all candidates of the connected regional patterns for Z, and we may calculate $\log \lambda(Z)$ for each candidate to find the optimal solution which gives the highest value. However, it is too time-consuming to generate all possible candidates because the number of connected regional patterns grows exponentially with respect to the number of regions m. As mentioned in the introduction section, most of existing methods put some practical restrictions in shape or size of the cluster candidates to trade off the exactness and computation time, for example, using circular-shaped window scan [10], exhaustive search bounded by a given size k [16], and a bounded search based on Echelon analysis [11, 7]. Anyway, those existing methods have a limitation that they can find the optimal solution only from the restricted subset of cluster candidates.

After finding the hotspot cluster, it is another important task to assess the statistical significance of the result whether it is incidental or somehow biased. In general, combinatorial hypothesis assessment is a hard problem because it requires a multiple-testing procedure. So far, no good methods are known to compute the exact p-value of the hotspots under multiple-testing correction, so the Monte Carlo method has been used commonly in the literature. The method generates a random data with the same occurrence ratio under null hypothesis and applies the hotspot detection to the random data. Repeating this process for many times (typically 9,999 times), and the p-value of the hotspot cluster can be evaluated to see the ranking in the distribution of the maximum statistic values for the random data. Although the Monte Carlo method gives asymptotically correct p-values as the number of random repetitions, the computation time grows linearly. Even if it takes one minute to execute hotspot detection, repeating the process 9,999 times requires about one week. Thus, the computation time is an important factor in the methods of hotspot detection.

3 Combinatorial Hotspot Mining

Maximizing $\log \lambda(Z)$ and checking the connectivity of the regions are very different properties, so it becomes difficult to deal with the two requirements together. In this work, we propose to explore all possible combinations of regional patterns for Z allowing the non-connected patterns. Here we call this problem *combinatorial hotspot mining*, to enumerate all combinatorial regions which are more significant than a given threshold of the statistic value θ. In this problem setting, the search space becomes as much as the power set of the regional items; however, the problem structure becomes simpler than original one.

If we could enumerate all combinatorial hotspot patterns which are more significant than θ, then the enumerated set must include all the connected hotspot patterns, and we may explore the connected ones after generating all combinatorial hotspots. Clearly, the maximum statistic value for the combinatorial hotspot patterns can be used as an upper bound of the maximum statistic value for the connected patterns. We can also expect that if the observed data is strongly biased by some geographical factors, then nearly connected patterns may appear in the optimal or nearly optimal solutions.

In our problem setting, no hotspot solutions may be found if the threshold θ is very high, but anyway, the maximum statistic value $\log \lambda(Z^*)$ can be obtained in the exploration process. On the other hand, an exponentially large number of solutions may be enumerated for a small threshold θ. It is practically important to find an appropriate θ, and it can be done in the following way.

1. Once execute hotspot mining giving a very high θ, and find the maximum statistic value $\log \lambda(Z^*)$.

2. Execute the same procedure with θ slightly lower than $\log \lambda(Z^*)$, then a few solutions are found.

3. Repeat the procedure by decreasing θ, and stop when an appropriate number of solutions are obtained.

Such usage is very similar to *frequent itemset mining* [17]. For the frequent itemset problem, a very fast, output-linear time algorithm has been known, and one may find an appropriate threshold frequency very quickly. However, the hotspot mining is still difficult problem even if we don't consider the connectivity in Z. One may consider a simple method that we merge a region into the cluster one by one from the highest occurrence ratio to the lowest, and we stop when the statistic value got a peak. However, the spatial scan statistic $\lambda(Z)$ has a nonlinear behavior, thus multiple peaks may be observed in such a simple method. We have to check various combinatorial patterns for exact enumeration, so it may require an exponential computation time with respect to the number of regions m.

For this problem, the authors' group conducted preliminary work [5], and we observed that a naive exhaustive search takes several hours when $m = 40$. We need something novel ideas to accelerate hotspot detection for applying to the practical data with larger m.

4 Proposed Algorithm

4.1 Basic Back Tracking Algorithm

First we show a naive back tracking algorithm as follows. Here $tail(Z)$ represents the last region-ID included in Z. The procedure is invoked by **BackTrack**(ϕ, θ), and finally it returns a number of hotspot patterns which are more significant than the threshold θ. max is a global variable to store the maximum statistic value in all combinatorial patterns.

```
ALGORITHM BackTrack ($Z, \theta$)
global variable: $max$ (initialized $max \leftarrow 0$)
1. $c \leftarrow 0$
2. if $\log\lambda(Z) > max$ then $max \leftarrow \log\lambda(Z)$
3. if $\log\lambda(Z) \geq \theta$ then output $Z$, $c \leftarrow 1$
4. for each region $e > tail(Z)$,
       $c \leftarrow c +$ BackTrack ($Z \cup \{e\}, \theta$)
5. return $c$
```

This naive algorithm explores the combinatorial space exhaustively, so always $O(2^m)$ times of recursive calls are executed, no matter how many solutions are found. In order to accelerate the algorithm, we may prune a search space by omitting the recursive call if some good estimation measure tells that it is no chance to be more significant than the threshold θ. Unfortunately it is not clear how to calculate such an estimation measure because the spatial scan statistic $\log\lambda(Z)$ is nonlinear and it cannot be simply decomposed.

4.2 Main Idea of Pruning

We found that the spatial scan statistic has partial monotonicity as follows.

THEOREM 4.1. *Suppose $Z, Z' \subseteq G$.*

(1) If $c(Z) = c(Z')$ and $n(Z) \geq n(Z')$, then $\lambda(Z) \leq \lambda(Z')$.

(2) If $n(Z) = n(Z')$ and $c(Z) \leq c(Z')$, then $\lambda(Z) \leq \lambda(Z')$.

Proof. Look at the upper part of the formula of $\lambda(Z)$ and rename $n(Z), c(Z)$ as x, y, respectively, then it can be expressed as:

$$f(x,y) := \left(\frac{y}{x}\right)^y \left(\frac{c(G)-y}{n(G)-x}\right)^{(c(G)-y)}.$$

Here, we may prove the following properties.

(1) If y is fixed, $f(x,y)$ monotonically decreases for x,

(2) If x is fixed, $f(x,y)$ monotonically increases for y,

under the λ's condition: $\dfrac{y}{x} > \dfrac{c(G)-y}{n(G)-x}$.

Proof for (1):

Since $f(x,y)$ is always positive, we may prove the monotonicity of $F(x,y) = \log f(x,y)$ as:

$$F(x,y) = y\log\frac{y}{x} + (c(G)-y)\log\frac{c(G)-y}{n(G)-x}.$$

Thus, the partial derivative of $F(x,y)$ with respect to x is obtained as:

$$\frac{\partial}{\partial x}F(x,y) = \frac{-n(G)\cdot y + c(G)\cdot x}{y(n(G)-x)}.$$

Here, the λ's condition $\dfrac{y}{x} > \dfrac{c(G)-y}{n(G)-x}$ implies $n(G)\cdot y > c(G)\cdot x$, thus $\frac{\partial}{\partial x}F(x,y)$ is always negative, and we can see $F(x,y)$ monotonically decreases for x.

Proof for (2):

As well as (1), we will show the monotonicity of $F(x,y)$ for y. The partial derivative of $F(x,y)$ with respect to y is obtained as:

$$\frac{\partial}{\partial y}F(x,y) = \log\frac{y}{x}\frac{n(G)-x}{c(G)-y}.$$

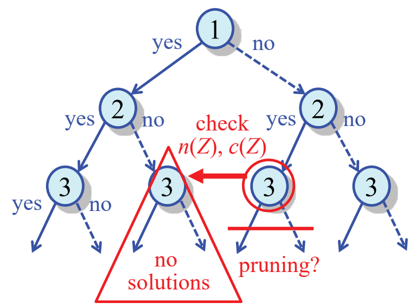


Figure 1: Back track search and pruning

Here, the λ's condition $\dfrac{y}{x} > \dfrac{c(G)-y}{n(G)-x}$ implies $\dfrac{y(n(G)-x)}{x(c(G)-y)} > 1$, thus $\frac{\partial}{\partial y}F(x,y)$ is always positive, and we can see $F(x,y)$ monotonically increases for y. □

From Theorem 4.1 (1) and (2), we get the following beautiful property of $\lambda(Z)$.

THEOREM 4.2. *Suppose* $Z, Z' \subseteq G$.
If $n(Z) \geq n(Z')$ *and* $c(Z) \leq c(Z')$, *then* $\lambda(Z) \leq \lambda(Z')$.

Proof. Let us consider a virtual region Y such that $n(Z) = n(Y) \geq n(Z')$ and $c(Z) \leq c(Y) = c(Z')$. Then, Theorem 4.1 (1) shows $\lambda(Y) \leq \lambda(Z')$, and Theorem 4.1 (2) shows $\lambda(Z) \leq \lambda(Y)$. Therefore, we can see $\lambda(Z) \leq \lambda(Z')$. □

Here we can see the useful property for the search space pruning.

THEOREM 4.3.
Suppose $Z, Z', Z^r \subseteq G$ *and* $Z^r \cap (Z \cup Z') = \emptyset$.
If $n(Z) \geq n(Z')$ *and* $c(Z) \leq c(Z')$, *then*
$\lambda(Z \cup X) \leq \lambda(Z' \cup X)$ *for any* $X \subseteq Z^r$.

Proof. It is clear because
$n(Z) \geq n(Z')$ implies $n(Z \cup X) \geq n(Z' \cup X)$ and
$c(Z) \leq c(Z')$ implies $c(Z \cup X) \leq c(Z' \cup X)$. □

From the above observation, we can design the search space pruning as follows. Assume that we get a combinatorial set of regions Z during the back track search as shown in Fig. 1, and we have already explored another set Z' such that $tail(Z) = tail(Z')$, but there were no significant solutions in all the descendants of Z'. In such a situation, if we found $n(Z) \geq n(Z')$ and $c(Z) \leq c(Z')$, then it tells that we have no chance to find a significant solution in the descendants of Z, and also no chance to improve the maximum statistic value max. In such cases, we can safely omit the further recursive call to prune the search space.

We note that this pruning idea works correctly if we explore all combinatorial regions. If we have some restrictions in shape or size of the clusters, Theorem 4.3 does not always stand, and the pruning method may not work correctly. We will discuss some ideas of conditional hotspot mining in Section 4.4.

4.3 Implementation of Pruning Function

Here we describe pseudo code of **BackTrack_Prune** equipped with the pruning function. In this algorithm, if no solutions are found after the recursive calls from the point of Z, then **EntPrune**() stores the parameters $n(Z), c(Z)$. **ChkPrune**() checks the parameters at the point of Z and tells whether pruning is possible or not.

```
ALGORITHM BackTrack_Prune $(Z, \theta)$
global variable: $max$ (initialized $max \leftarrow 0$)
1. $c \leftarrow 0$
2. if ChkPrune($n(Z), c(Z), tail(Z)$) then return $c$
3. if $\log\lambda(Z) > max$ then $max \leftarrow \log\lambda(Z)$
4. if $\log\lambda(Z) \geq \theta$ then output $Z$,  $c \leftarrow 1$
5. for each region $e > tail(Z)$,
       $c \leftarrow c +$ BackTrack_Prune $(Z \cup \{e\}, \theta)$
6. if $c = 0$ then call EntPrune($n(Z), c(Z), tail(Z)$)
7. return $c$
```

Then, we discuss the implementation how to store and check the parameters for pruning.

The property shown in Theorem 4.2 seems very simple; however, the implementation is not so easy because we have to deal with the two dimensional parameters together. If it was one dimensional, we may simply record the highest score of the parameter for each $tail(Z)$. In our problem, as illustrated in Fig 2, we should record a set of the maximal (Pareto-optimal) points, and we should check whether the current Z is dominated by at least one of the maximal points or not. The total number of the maximal points may grow exponentially large with respect to m, so it might consume time and space a lot, even more than the saving effect of the pruning.

For storing and looking up such two-dimensional numerical data, there have been a lot of work in the field of *geometric range search* [1], and basically, balanced binary search trees and some variations are commonly used. It is known that the computational cost for each insertion, lookup, and removal is roughly $O(\log n)$ when n items are stored. We may utilize *std::map* in the standard C++ library for this purpose.

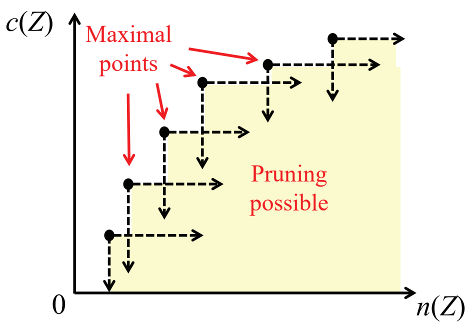


Figure 2: Maximal (Pareto-optimal) points

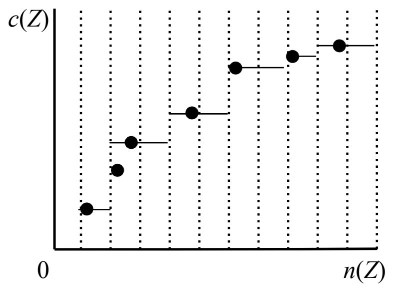


Figure 3: Area slicing method

Our pruning strategy is considered for accelerating the search process, so there is no change in correctness of the solutions if we overlooked some chances of pruning. Even in such cases, the pruning condition must be satisfied at the next recursion step, so anyway the search space will be pruned. Thus, a small probability of "false negative" is allowed in checking the pruning condition. From this observation, we implemented a simple alternative method based on area slicing for handling the maximal points. As shown in Fig. 3, we partition the two dimensional space into k slices (k is a given constant) with respect to $n(Z)$, then we record the highest score of $c(Z)$ for each slice. We prepare such a highest score table for each $tail(Z)$, so in total $k \cdot m$ variables are used. This method features the simple data structure and a constant time operation for each insertion and lookup.

The number of slices k is a key to the efficiency of this method. Too large k wastes not only the memories but also the time to maintain the highest score for each slice. Too small k may overlook too many chances of pruning, then computation time grows exponentially. The optimal k depends on the input data.

Figure 4 illustrates a typical example of the distribution of $(n(Z), c(Z))$ over the contour lines of $\lambda(Z)$.

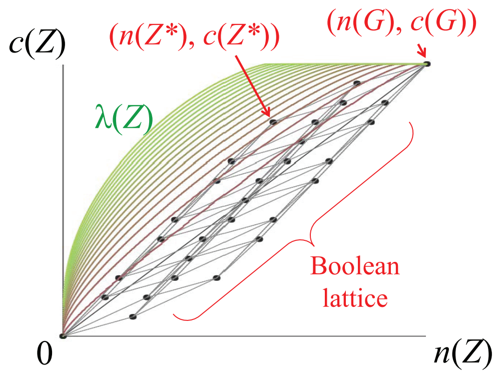


Figure 4: Distribution of $(n(Z), c(Z))$ over the contour lines of $\lambda(Z)$

We can observe that the points of the combinatorial patterns form a Boolean lattice in the two dimensional space. As the depth of the Boolean lattice is m, it is clear that at least m slices are required for well handling the maximal points, and more fine slices will make more surely pruning. In our implementation, we empirically set $k = 64m$. In this setting, the area slicing method is several times faster than a standard range search method using binary search trees, in many cases of our experiments.

4.4 Conditional Hotspot Mining

In a small modification of the above algorithm, we can enumerate a subset of hotspot clusters bounded by population or occurrence.

Bounded by Population

We may easily put an additional condition to enumerate relatively small hotspot clusters such that $\log \lambda(Z) \geq \theta$ and $n(Z) \leq n_{max}$, where n_{max} is a given upper-bound in terms of population. If the pruning condition of Theorem 4.2 is satisfied, $n(Z) \geq n(Z')$ always stands. In this case, if Z' is not a solution due to the upper-bound n_{max}, then Z also violates n_{max}, so anyway Z is not a solution. Thus, we don't need any special treatment in the pruning process. Unfortunately, it is not easy to enumerate the larger hotspot clusters in terms of population because the above discussion does not work.

Bounded by Occurrence

This is a duality that we can easily enumerate relatively large hotspot clusters such that $\log \lambda(Z) \geq \theta$ and $c(Z) \geq c_{min}$, where c_{min} is a given lower-bound in terms of occurrence. If the pruning condition of Theorem 4.2 is satisfied, $c(Z) \leq c(Z')$ always stands. In this case, if

Table 2: Experimental result for 47 pref. dataset

thres. θ	#solution	total calls	time(sec)
117.0	0	9,847	0.05
116.0	5	10,130	0.05
110.0	4,414	21,979	0.05
100.0	696,559	988,933	0.2
90.0	22,049,421	27,381,848	4.1
85.0	92,666,978	111,602,318	14.9
80.0	341,003,096	401,037,972	49.8

Z' is not a solution due to the lower-bound c_{min}, then Z also violates c_{min}, so anyway Z is not a solution. Thus, we don't need any special treatment in the pruning process. Similarly, it is not easy to enumerate the smaller hotspot clusters in terms of occurrence.

5 Experiments and Evaluations

We implemented the above algorithm, named "power set scan", and conducted experimental evaluation. Our program consists of about 600 lines of C++ code. We used Panasonic Let's Note CF-SZ6 (Intel Core i7, 2.7GHz, 8GB memory) and GNU C++ 7.3.0 on Cygwin 2.10.0, Windows 10. Our program has options for just counting the significant hotspot patterns, or actually outputting the solutions to a file. In the following experiments, we show the computation time in the case of the counting mode. In our implementation, we first sort the regions of the input data in terms of occurrence ratio from the highest to the lowest, then start the back track search for all combinatorial regions in a lexicographical order.

In our experiments, we used the three real datasets, as follows.

- The number of suicides in the 47 prefectures of Japan in 2006 [14]. ($m = 47$, $n(G) = 128,066,211$, $c(G) = 21,897$)
- Occurrence of SIDS (Sudden Infant Death Syndrome) in the 100 counties of North Carolina [3]. ($m = 100$, $n(G) = 752,354$, $c(G) = 1,503$)
- Occurrence of leukaemia in Upstate New York [10]. ($m = 281$, $n(G) = 1,057,673$, $c(G) = 574$)[2]

First, we conducted a series of experiments for the 47 Japanese prefectures dataset to enumerate the significant solutions for varying the threshold θ. Table 2 shows that the computation time is only less than 0.1 second when the number of solutions is not large. When $\theta = 117$, no solutions are found, but in this process we can compute the maximum statistic value $\log \lambda(Z^*) = 116.341$, and we found the optimal Z^* which consists of 22 regions. It is a remarkable result that we could obtain the exact maximum value of the spatial scan statistic for this dataset scale of 47 regions, without any restriction in shape or size of clusters.

[2]The original data has $c(G) = 592$. The difference is because the original statistical data includes some decimal part numbers, and we rounded them into integers.

Table 3: Comparison with a naive exhaustive search

m	naive search		proposed method	
	total calls	time(sec)	total calls	time(sec)
20	1,048,575	0.07	1,230	0.04
25	33,554,431	0.6	1,820	0.04
30	1,073,741,823	17.9	2,717	0.05
35	34,359,738,367	560.0	5,209	0.05
40	–	time out	7,103	0.05
45	–	time out	9,059	0.05
47	–	time out	9,847	0.05

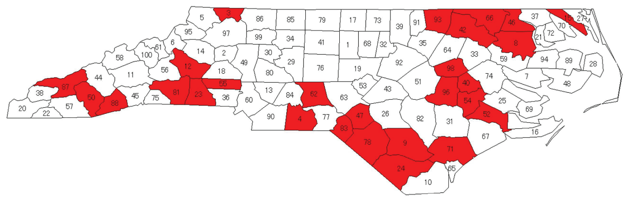
Table 4: Experimental result for SIDS dataset

thres. θ	#solution	total calls	time(sec)
68.0	0	9,426	0.07
67.7	2	9,718	0.07
67.5	41	9,904	0.07
67.0	1,582	13,035	0.07
66.5	19,850	37,266	0.08
66.0	152,525	196,057	0.2
65.5	901,043	1,051,017	0.6
65.0	4,437,311	5,058,184	2.6

Table 3 shows another series of experiments to compare the computation time between a naive exhaustive search and our fast algorithm. We put a sufficiently large threshold θ to compute only the maximum statistic value. The naive search is too time-consuming for the 47 regions dataset, so we extracted the first m regions from the original data. The experimental results show the contrastive effect of our pruning method. It is estimated that the naive search takes about one month to complete the exploration for the 47 regions. We can observe that our power set scan algorithm runs more than a million or more times faster than the naive search in this dataset.

Next, we applied our algorithm to the SIDS dataset with 100 regions, to enumerate significant solutions by varying the threshold θ. The results are shown in Table 4. SIDS dataset is the well-known spatial dataset in the hotspot analysis research community, but

First ranked significant pattern ($\log \lambda = 67.720$)



Second ranked significant pattern ($\log \lambda = 67.711$)

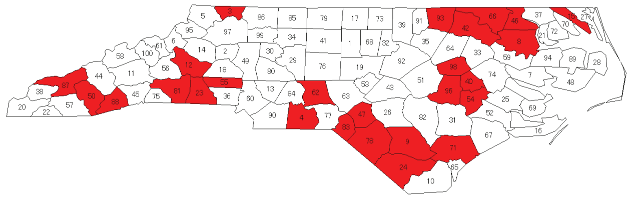


Figure 5: Top two significant patterns for SIDS dataset

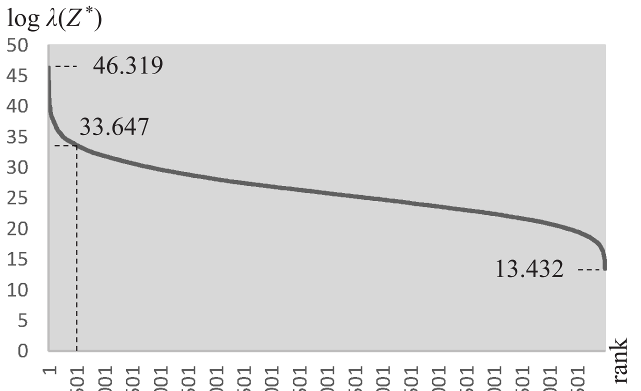


Figure 6: Distribution of $\log \lambda(Z^*)$ for 9,999 cases of randomized datasets.

Table 5: Experimental result for New York dataset

thres. θ	#solution	total calls	time(sec)
143.0	0	10,796	0.1
142.5	1	11,250	0.1
142.3	130	11,486	0.1
142.0	4,995	17,251	0.1
141.8	30,595	45,654	0.1
141.5	319,199	364,459	0.4
141.3	1,293,307	1,399,955	1.1
141.0	8,845,457	9,312,582	6.1

no one has reported the exact maximum value of the spatial scan statistic for all possible patterns. This is the first obtained knowledge. Figure 5 shows the top two significant patterns in the SIDS dataset. We can observe that the top significant pattern contains many non-connected regions. Anyway, we can say that $\log \lambda(Z^*) = 67.720$ is an upper-bound for any connected hotspot cluster in this dataset.

Let us evaluate the p-value of the top significant hotspot pattern whether it is incidental or somehow biased. For this assessment, we generated 9,999 cases of randomized datasets with the same occurrence ratio as the original SIDS dataset under the null hypothesis. We then executed our power set scan algorithm for each random dataset. Our program took only 96.7 seconds in total for executing 9,999 cases of the randomized dataset. Figure 6 shows the result that the maximum statistic $\log \lambda(Z^*)$ for the 9,999 random datasets were distributed from 13.432 to 46.319, and the top 500-th ranked value was 33.647. This means that the observation data is significantly biased with 5% p-value if $\log \lambda(Z^*) > 33.647$. Our original observation $\log \lambda(Z^*) = 67.720$ is much higher than the top 0.01% rank of random execution, thus the null hypothesis is clearly rejected in very high confidence. Our fast algorithm will greatly contribute to drastic acceleration of the Monte Carlo method for computing the p-values of the hotspot clusters.

Finally we applied our algorithm to the dataset of leukaemia in Upstate New York. Although this dataset contains as many as 281 regions, Table 5 shows that our algorithm still works well. We got the maximum statistic $\log \lambda(Z^*) = 142.503$, and we found the optimal Z^* which consists of 116 regions. In the previous work, for example, Flexible scan [16] works well only for small to moderate cluster size, up to 20 or 30. Our result of finding the 116 regions pattern demonstrates the outstanding scalability of our algorithm.

6 Summary and Future Work

In this work, we presented a very fast algorithm "power set scan" to enumerate all combinatorial regions which are more significant than a given threshold value. Our algorithm features the fast exploration by pruning the search space based on the partial monotonicity of the spatial scan statistic. Experimental results for the practical datasets show that our method works a million or more times faster than the previous naive search method. Our method works practically for a dataset with several hundreds of regions, and it will drastically accelerate hotspot analysis in various fields.

In our discussion, we assumed Poisson model for event occurrences, but some other models, such as Bernoulli model or chi-squared model, may also have the similar partial monotonic property so that Theorem 4.2 stands. We may apply the same algorithm for the hotspot mining based on those different models.

Our algorithm enumerates all combinatorial hotspot patterns which are more significant than a threshold θ. If thousands of hotspot clusters are shown, it may be hard for human beings to understand the meaning of the hotspot patterns. Some ideas of data visualization could be considered, for example, a heat map representation would be helpful to accumulate a large number of hotspot clusters.

Another feature of our method is that it covers all combinatorial regions including non-connected patterns. This problem setting makes pruning easier, but in some real applications, one may be interested in the connected patterns only. In such cases, we should consider the way to extract the connected clusters from a large number of combinatorial patterns. The (Zero-suppressed) Binary Decision Diagram-based indexing techniques [8, 6] would be useful for enumerating the connected hotspot patterns, and developing those techniques is in our future work.

In our real-life, there are so many statistical data which consist of a number of partitions, each of which has population and occurrence. Categorizing all partitions into one of the two groups is a fundamental and important analysis in various fields, such as bioinformatics, epidemiology, business, agriculture, engineering, transportation, education, etc. We hope our fast algorithm will be utilized widely in those applications.

References

[1] Pankaj K. Agarwal and Jeff Erickson. Geometoric range searching an its relatives. In *the 1996 AMS-IMS-SIAM Joint Summer Research Conference, Discrete and Computational Geometry–Ten Years Later*, pages 1–56, 1999.

[2] L. Anselin. Local indicators of spatial association-LISA. *Geographic Analysis*, 27(2):93–115, 1995.

[3] N. A. Cressie and N. Chan. Spatial modeling of regional variables. *Journal of The American Statistical Association*, 84(406):393–401, 1989.

[4] J. Cuzick and R. Edwards. Spatial clustering for inhomogeneous populations. *Journal of the Royal Statistical Society, Series B*, 52(1):73–104, 1990.

[5] Ryo Ishimaru. *A Study on ZDD-Based Representation for Hotspots of Statistical Data.* Master Thesis of Graduate School of Information Science and Technology, Hokkaido University (in Japanese), 2018.

[6] Fumio Ishioka, Jun Kawahara, Masahiro Mizuta, Shin-ichi Minato, and Koji Kurihara. Evaluation of hotspot cluster detection using spatial scan statistic based on exact counting. *Japanese Journal of Statistics and Data Science, Springer*, 2(1), 2019. (in press)

[7] Fumio Ishioka, Koji Kurihara, Hiroshi Suito, Yasuo Horikawa, and Yoshiro Ono. Detection of hotspots for three-dimensional spatial data and its application to environmental pollution. *Journal of Environmental Science for Sustainable Society*, 1:15–24, 2007.

[8] Jun Kawahara, Takashi Horiyama, Keisuke Hotta, and Shin-ichi Minato. Generating all patterns of graph partitions within a disparity bound. In *Proc. of the 11th International Workshop of Algorithms and Computation (WALCOM2017), (LNCS 10167, Springer)*, pages 119–131, 2017.

[9] Martin Kulldorff. A spatial scan statistic. *Communications in Statistics: Theory and Methods*, 26(6):1481–1496, 1997.

[10] Martin Kulldorff and N. Nagarwalla. Spatial disease clusters: Detection and inference. *Statistics in Medicine*, 14(8):799–810, 1995.

[11] Koji Kurihara. *Classification of geospatial lattice data and their graphical representation*, pages 251–258. Springer, 2004.

[12] P. A. P. Moran. The interpretation of statistical maps. *Journal of the Royal Statistical Society, Series B*, 10(2):243–251, 1948.

[13] Joseph I. Naus. The distribution of the size of the maximum cluster of points on a line. *Journal of the American Statistical Association*, 60(310):532–538, 1965.

[14] National Police Agency of Japan. Suicide situation during 2016, https://www.npa.go.jp/publications/statistics/safetylife/jisatsu.html, 2016.

[15] Toshiro Tango. A class of tests for detecting "general" and "focuses" clustering of rate diseases. *Statistics in Medicine*, 14(21-22):2323–2334, 1995.

[16] Toshiro Tango and Kunihiko Takahashi. A flexibly shaped spatial scan statistic for detecting clusters. *International Journal of Health Geographics*, 4:11, 2005.

[17] T. Uno, M. Kiyomi, and H. Arimura. LCM ver.2: efficient mining algorithms for frequent/closed/maximal itemsets. In *Proc. IEEE ICDM'04 Workshop FIMI'04 (International Conference on Data Mining, Frequent Itemset Mining Implementations)*, 2004.