Vol.2018-AL-169 No.3
2018/9/3

スキャン統計量に基づく組合せホットスポット抽出を行う高速アルゴリズム

湊 真一[1,a)]　川原 純[2,b)]　水田 正弘[3,c)]　石岡 文生[4,d)]　栗原 考次[4,e)]

概要：例えば都道府県のような地理的区画に分けられた統計データに対して，特定の事象が集中的に発生している領域を検出する問題はホットスポット検出問題と呼ばれ古くから研究されている．既存研究では，発生の集中度を表す指標として，空間スキャン統計量と呼ばれる対数尤度比が広く用いられている．本研究では，連結とは限らないすべての区画の組合せについて，事前に与えた閾値よりも高いスキャン統計量となる組合せを全て列挙する超高速なアルゴリズムを提案する．本手法では，スキャン統計量の部分的な単調性を利用した枝刈りにより高速な探索を実現しており，実験の結果，47 都道府県の例題に対して，素朴な方法に比べて 100 万倍以上高速に，統計量最大のホットスポットを抽出することが可能となった．本手法は 100〜200 区画程度までの統計データに対して，現実的な計算時間で動作する．

キーワード：ホットスポット検出，スキャン統計量，組合せ列挙

A Fast Algorithm for Combinatorial Hotspot Mining Based on Scan Statistic

MINATO Shin-ichi[1,a)]　KAWAHARA Jun[2,b)]　MIZUTA Masahiro[3,c)]　ISHIOKA Fumio[4,d)]
KURIHARA Koji[4,e)]

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 used widely 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 data 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 statistical data with up to 100 or 200 regions.

Keywords: Hotspot detection, Scan statistic, Combinatorial enumeration

[1] 京都大学
Kyoto University, Kyoto, 606-8501, Japan
[2] 奈良先端科学技術大学院大学
NAIST, Ikoma, 630-0192, Japan
[3] 北海道大学
Hokkaido University, Sapporo, 060-0814, Japan
[4] 岡山大学
Okayama University, Okayama, 700-8530, Japan
[a)] minato@i.kyoto-u.ac.jp
[b)] jkawahara@is.naist.jp
[c)] mizuta@iic.hokudai.ac.jp
[d)] fishioka@okayama-u.ac.jp
[e)] kurihara@ems.okayama-u.ac.jp

1. はじめに

都道府県や市町村のような地理的区画に分けられた統計データに対して，病気や事故等の事象が集中的に発生しているような領域（ホットスポット）を検出する問題は，様々な社会的現象の解析に用いられる基本的かつ重要な問題であり，古くから研究されている．例えば，空間的自己相関の観点からホットスポットを検出する手法 [1] や，全領域の中を一定の規則に基づいた小領域で走査 (スキャン) し

表 1　入力データの例

区画番号	人口	発生数
1（北海道）	5,401,210	1,004
2（青森県）	1,338,465	279
3（岩手県）	1,289,470	322
...	...	...
47（沖縄県）	1,461,231	258

ていき，ホットスポットを検出する手法 [2], [7] などが提唱されている．そうした中，ホットスポット検出のための優れた指標の一つに，1997 年に Kulldorff が提案した空間スキャン統計量 [6] がある．これは，元々は，時系列データをスキャンしたときに発生頻度が高い区間を検出するための尤度比検定統計量を，空間データのホットスポット検出問題に拡張したものであり，統計的な性質の良さから近年広く用いられている．空間データでは，ホットスポット候補となる領域の組合せ方が区画数に対して指数関数的に増大するため，領域の形や大きさを限定してスキャンを行う方法が研究されてきた．Kulldorff が提案した手法は同心円状に領域をスキャンするため，円形状のホットスポットしか検出できない．この問題を改善するための手法として，最大ブロックサイズを決めて総当り的に領域のパターンをスキャンする Flexible スキャン法 [8] や，Echelon 解析により得られるデータの階層構造に基づいて領域のパターンをスキャンする方法 [4], [10] 等，いくつかの手法が提案されている．

本研究では，空間スキャン統計量に基づく探索を行う際に，形や大きさを限定した連結領域をスキャンするのではなく，非連結領域も許した全区画のべき集合の中から，一定の閾値を超えるスキャン統計量を持つような区画の組合せを全て抽出し列挙するという「組合せホットスポット抽出問題」を考える．この問題では，ホットスポット候補の数はさらに膨大になるが，探索空間は地理的な制約のない単なるべき集合となるため，問題の構造はむしろ単純になる．我々は，空間スキャン統計量に部分的な単調性があることを見出し，その性質を利用した探索空間の枝刈りを用いて非常に高速な厳密解探索アルゴリズムを開発した．実験の結果，47 都道府県の例題に対して，素朴な方法に比べて 100 万倍以上高速に，統計量最大のホットスポットを抽出することが可能となった．本手法はおよそ 100〜200 区画程度までの統計データに対して現実的な計算時間で動作することから，今後，様々な分野への応用が期待される．

以下の本文では，2 章でホットスポット検出とスキャン統計量についての準備を行い，3 章で問題設定を行う．4 章で提案手法を説明し，5 章で実験結果を示す．6 章でまとめと今後の課題を述べる．

2. ホットスポット検出とスキャン統計量

2.1 想定する統計データ

本研究では，都道府県や市町村のような地理的区画に分けられた統計データを入力とする．表 1 に簡単な例を示す．各区画データは区画番号を付与されており，人口と発生数の 2 つの数値データを持つ[*1]．より形式的には，m 個の区画に分けられた領域集合 $G = \{1, 2, \ldots, m\}$ に関する数値データ $\mathcal{D} = \{(n_1, c_1), (n_2, c_2), \ldots, (n_m, c_m)\}$ を入力とする．各区画 i の数値 $(n_i, c_i) \in \mathbf{N^2}$ は，n_i が人口（母集団の数），c_i が発生数（属性を持つものの数）を表し，$0 \leq c_i \leq n_i$ である．発生率は c_i/n_i で表される．

$Z \subseteq G$ となる部分領域 Z を取り出した時，その領域の人口と発生数を $n(Z), c(Z)$ と表記する．すなわち，

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

である．以下では Z の補領域を Z^c と書く．明らかに $n(Z^c) = n(G) - n(Z), \quad c(Z^c) = c(G) - c(Z)$ となる．

2.2 スキャン統計量

空間スキャン統計量は，ある領域内の地点に起きた現象が偶然によるものか否かを検定し，有意に高い地域群（ホットスポット）を検出するための尤度比検定統計量である．元々は，時系列データをスキャンしたときに発生頻度が高い区間を検出するために作られた統計量を，空間データのホットスポット検出問題に拡張したものであり，統計的な性質の良さから近年広く用いられている．

検定対象とする部分領域を Z とすると，Z の内部では各個人がある属性を持つ確率を p_1，外部の Z^c では確率を p_2 とする．属性を持つ確率は互いに独立であるとする．このとき，帰無仮説は $p_1 = p_2$，対立仮説は $p_1 > p_2$ である．

ポアソン分布を仮定すると，総発生数 $c(G)$ が観測値と一致する確率は，以下の式で表される．

$$\frac{\exp[-E[c(G)]](E[c(G)])^{c(G)}}{c(G)!}$$

ただし $E[c(G)] = p_1 n(Z) + p_2 n(Z^c)$ で総発生数の期待値である． ある地点 x での発生密度は，x での母集団の数を $n(x)$ とすると

$$\begin{cases} \dfrac{p_1 n(x)}{E[c(G)]} & \text{if } x \in Z \\ \dfrac{p_2 n(x)}{E[c(G)]} & \text{if } x \in Z^c \end{cases}$$

で表される．このときポアソンモデルに対する尤度関数は以下のように与えられる．

$$L(Z, p_1, p_2) = \frac{\exp[-E[c(G)]]}{c(G)!} p_1^{c(Z)} p_2^{c(Z^c)} \prod_{x_i \in G} n(x_i)$$

ここで，Z を固定したときの p_1, p_2 の最尤推定量として $\hat{p_1} = c(Z)/n(Z), \ \hat{p_2} = c(Z^c)/n(Z^c)$ を与えると，尤度関

[*1] 必ずしも住民に関するデータでなくても，例えば工場の生産台数と不良品の数でも良い．

数の最大値は

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

と表せる．一方，帰無仮説上で $p_1 = p_2$ を仮定した場合，尤度関数の最大値は Z に関わらず以下の式で表される．

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

$L(Z)$ と L_0 の比を $\lambda(Z)$ とすると，

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

と表される．これが空間スキャン統計量と呼ばれる統計量である．なお，$\lambda(Z)$ は Z と Z^c に関して対称な形となるため，Z の方が Z^c よりも発生率が高いという制約条件が付与されている．空間スキャン統計量 $\lambda(Z)$ を最大にするような Z は，領域集合が，異なる 2 種類の発生確率を持つ 2 群に分かれると仮定したときに，観測された総発生数を最も高い尤度で説明できる分割であるという理論的な根拠を持つ．実用的観点から見ても，ブロックサイズが大き過ぎても小さ過ぎても尤度比は大きくならず，ちょうど良いブロックサイズのときに最大値となるため，事前にブロックサイズを考慮しなくても，1 種類の統計量だけに基づいてホットスポットを求められるという特長を持つ．実際に計算をする場合は，対数を取って $\log\lambda(Z)$ を計算し，これを最大化する Z を探索する方が容易に計算できる．統計量の計算コストはそれほど大きくはないが，べき乗（または対数）が含まれるため，組合せ最適化問題のコスト関数として見たときには，上界や下界の分析は簡単ではない．

2.3 ホットスポットの探索

空間スキャン統計量 $\lambda(Z)$ が最大となる Z を探索するのがホットスポット検出問題である．基本的には解候補となる連結な領域パタンを逐次生成し，それぞれの統計量を計算して，最も大きくなる候補を探せばよいが，領域パターン数は区画数に対して指数関数的に増大するため，少し規模が大きくなると計算時間がかかり過ぎる．元々は，時系列データをスキャンして発生頻度が高い区間を検出するために開発された方法を空間データに拡張したものであったため，領域の形や大きさを限定してスキャンを行う方法が研究されてきた．例えば，同心円状に領域をスキャンする Kulldorff の方法 [6]，最大ブロックサイズの範囲で網羅的にスキャンする Flexible スキャン法 [8]，Echelon 解析によるデータの階層構造に基づいてスキャンする方法 [4], [10] 等，いくつかの手法が提案されている．いずれにしても，候補に現実的な制約を与えて探索空間を絞り込んでいるため，その制約された範囲での最適解しか見つけられないという課題があった．

一方で，得られたホットスポットに有意性があるかどうかを検定する方法も重要である．今回の探索のように，同じデータに対して多数の仮説を探索して最も良いものを取る場合は，多重検定補正を行う必要があるが，今のところ多重検定補正を正しく行い p 値を直接求める良い方法は見つかっていないため，伝統的にモンテカルロ法が用いられている．これは，帰無仮説に基づきランダムデータを 999 通り生成して同じ方法で統計量の最大値を求めておき，観測データに対する結果が合計 1000 回のうち上位何%に入っているかで p 値を計算する方法である．モンテカルロ法は試行回数を増やせば真の p 値に収束することが保証されているが，1000 通り試行すれば 1000 倍の計算時間がかかるので，1 回当たり数秒だとしても数千秒かかってしまうという問題があり，少し大規模な問題になると適用が難しくなる．

3. 問題設定

3.1 非連結な領域も含む組合せホットスポットの全列挙

スキャン統計量 $\lambda(Z)$ が大きいということと，Z が飛び地のない連結領域であることは，全く性質が異なる事柄であり，それらを同時に扱おうとすると問題が難しくなる．本研究では，形や大きさを限定した連結領域をスキャンするのではなく，非連結領域も許した全区画 G のべき集合の中から，一定の閾値を超える $\lambda(Z)$ を持つような組合せを全て抽出し列挙するという「組合せホットスポット抽出問題」を考える．この問題では，ホットスポット候補の数はさらに膨大になるが，探索空間は地理的な制約のない単なるべき集合となるため，問題の構造はむしろ単純になる．

もしも，G のべき集合の中から，一定基準以上の統計量となるような区画組合せを全列挙できれば，その基準を超える連結ホットスポットはその中に含まれているはずであるから，連結ホットスポットを探したければその中から絞り込めば良い．少なくとも本手法によって，連結ホットスポットの統計量の上界を知ることができる．さらに，そもそも地理的要因で発生に偏りが生じているのであれば，本手法によってほぼ連結なホットスポットが上位に現れるはずである．

今回の問題設定では，十分大きな閾値を与えると解なしとなるが，その探索の過程で統計量の最大値が副次的に求まる．一方，閾値を小さくしていくと解の個数は指数関数的に増大し，閾値を 0 にすればべき集合の要素全てが解となる．適切な閾値を与えることが実用上は重要となるが，それは以下の手順で行える．

(1) まず十分大きな閾値を与えて探索を一度実行し，統計量の最大値を求める．

(2) 次にその最大値より少し小さな閾値を与えて探索すると少ない個数の解が出力される．
(3) 閾値を徐々に下げて実行を繰り返し，適切な個数の解になったところで終了する．

このような使用法は，頻出パタンマイニング [9] と類似している．頻出パタンマイニングでは出力線形時間のアルゴリズムが知られているので，解なしであれば一瞬で終了するので，迅速に適切な閾値を定めることができる．ホットスポット探索でも，そのような高速アルゴリズムが開発できれば実用的な意義は大きい．

しかし，連結という制約を外したとしても，空間スキャン統計量が最大となる区画組合せを求めることは，依然として難しい問題である．もしも各区画が完全に等人口で区切られていれば，あらかじめ発生率が高い順に並べておいて，上位から順に加えて行き，統計量がピークになった時点で止めれば最適解が得られる．しかし各区画の人口が一定でない場合，必ずしも発生率の大小とスキャン統計量の大小が一致しないため，様々な組合せを試す必要が生じ，指数関数的な探索空間が生じることになる．

本研究のような非連結な組合せを含むホットスポット探索に関しては，昨年度，著者の研究室の学生だった石丸 [12] が取り組み予備的実験を行った結果がある．それによれば，素朴な全探索では，区画数が 30〜40 程度の規模が，現実的な時間で解ける限界であることがわかっている．それ以上の規模の問題に対しては，無駄な探索を枝刈りするなど，何らかの工夫が必要であるが，計算結果の正しさを保証できるような高速化方法は残念ながら見つかっていなかった．

4. 提案手法

4.1 二分探索と枝刈りの枠組み

まず，素朴に二分探索を行う基本的な全探索アルゴリズム **BackTrack** を以下に示す．ここで θ は対数尤度比の閾値，$tail(Z)$ は Z に含まれる最も番号の大きい要素を表す．プログラムは **BackTrack**(ϕ,θ) を呼び出すことで開始され，ヒットした解の個数を値として返す．グローバル変数 max には，探索した中での対数尤度比の最大値が記録される．

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

BackTrack は素朴に全探索を行うので，ヒットする解の個数に関わらず，区画数 m に対して 2^m 回の再帰呼び出しが常に発生する．これを高速化するため，探索途中の各状態で何らかの尺度を計算し，それ以上どんな区画を追加しても閾値を超えられないということが分かれば，再帰呼び出しを打ち切り探索空間を枝刈りするという仕組みを入れたい．問題はどうやって枝刈りの可否を正しく判定するかである．空間スキャン統計量 $\lambda(Z)$ は単純な線形和ではなく，発生率に関して単調ではないので，枝刈り判定は自明ではない．

4.2 枝刈り判定方法

我々は，空間スキャン統計量に部分的な単調性が存在することに着目した．

定理 4.1. $Z, Z' \subseteq G$ において $c(Z) = c(Z')$ と固定したとき，$n(Z) \geq n(Z')$ ならば $\lambda(Z) \leq \lambda(Z')$ である．

定理 4.2. $Z, Z' \subseteq G$ において $n(Z) = n(Z')$ と固定したとき，$c(Z) \leq c(Z')$ ならば $\lambda(Z) \leq \lambda(Z')$ である．

証明 $\lambda(Z)$ の分子のみを取り出して，$n(Z), c(Z)$ を実数変数 x, y におきかえると，

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

と表せる．このとき以下を示せばよい．
(1) y を定数としたとき $f(x,y)$ は x に関し単調減少．
(2) x を定数としたとき $f(x,y)$ は y に関し単調増加．
ただし (1)(2) ともに $\frac{y}{x} > \frac{c(G)-y}{n(G)-x}$ であるとする．

(1) の証明

$F(x,y) = \log f(x,y)$ とおいて，$F(x,y)$ が x に関して単調減少であることを示す．

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

であるから，これを x で偏微分すると

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

が得られる．ここで $\frac{y}{x} > \frac{c(G)-y}{n(G)-x}$ より $n(G)\cdot y > c(G)\cdot x$ が成り立つので，分子は常に負となり，$\frac{\partial}{\partial x}F(x,y) < 0$ が成り立つ．したがって，$F(x,y)$ は x に関して単調減少である．(1) の証明終わり．

(2) の証明

(1) と同様に $F(x,y)$ が y に関して単調増加であることを示す．$F(x,y)$ を y で偏微分すると

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

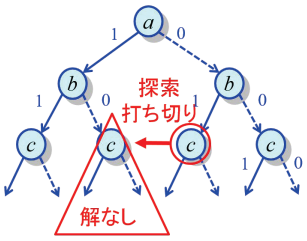


図 1　二分木探索と枝刈り

が得られる．ここで $\frac{y}{x} > \frac{c(G)-y}{n(G)-x}$ より $\frac{y(n(G)-x)}{x(c(G)-y)} > 1$ であるので，$\frac{\partial}{\partial y}F(x,y) > 0$ が成り立つ．したがって，$F(x,y)$ は y に関して単調増加である．(2) の証明終わり．□

空間スキャン統計量は，データを 2 群に分割したときの偏りの激しさを表すので，その意味から考えれば上記が成り立つのは自然である．塩水の中に濃度が濃い部分があったときに，その部分の塩分量を変えずに水分だけを薄い方から濃い方に移動させやれば濃度の偏りは減少する．一方，総重量を保ちながら塩分を濃い方から薄い方に移動させても濃度の偏りは減少する．

定理 4.1 および 4.2 より以下が成り立つ．

定理 4.3. $Z, Z' \subseteq G$ において，$n(Z) \geq n(Z')$ かつ $c(Z) \leq c(Z')$ ならば $\lambda(Z) \leq \lambda(Z')$ である．

証明　$n(Z) = n(Y) \geq n(Z')$ かつ $c(Z) \leq c(Y) = c(Z')$ であるような，ある仮想的な領域 Y を考える．これは $Y = Z'$ となるような Y を用意しておき，Y の塩分量は変えずに総重量が Z と同じになるように Y^c から Y に水分を移動させれば作ることができる．このとき，定理 4.1 より $\lambda(Y) \leq \lambda(Z')$ であり，定理 4.2 より $\lambda(Z) \leq \lambda(Y)$ であるから，$\lambda(Z) \leq \lambda(Z')$ が成り立つ．□

さらに，以下の定理が成り立つ．

定理 4.4. $Z, Z' \subseteq G$ において，Z, Z' のいずれにも含まれない残りの区画集合を Z^r とする．$n(Z) \geq n(Z')$ かつ $c(Z) \leq c(Z')$ ならば，任意の $X \subseteq Z^r$ に対して $\lambda(Z \cup X) \leq \lambda(Z' \cup X)$ である．

証明　$n(Z) \geq n(Z')$ ならば $n(Z \cup X) \geq n(Z' \cup X)$ であり，$c(Z) \leq c(Z')$ ならば $c(Z \cup X) \leq c(Z' \cup X)$ であることから明らか．□

以上の考察から，図 1 のような二分木状の探索において，探索途中にある組合せ Z に到達したとき，過去に探索済みの組合せのうち $tail(Z) = tail(Z')$ となるような Z' が存在し，もしも $n(Z) \geq n(Z')$ かつ $c(Z) \leq c(Z')$ で，なおかつ過去の Z' 以下の再帰呼び出しで解が 1 つも見つかっていなければ，Z 以下の再帰呼び出しで解が見つかる可能性がないことが分かる．また，Z 以下の再帰呼び出しでこれまでの統計量の最大値 max が改善される望みもない．したがって Z 以下の再帰呼び出しを打ち切って探索空間を枝刈りしても計算結果に影響はない．

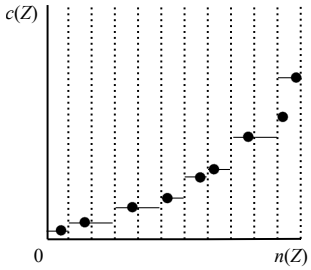


図 2　$n(Z)$ と $c(Z)$ の極大値の管理方法

4.3　枝刈り判定方法の実装

枝刈り判定を追加したアルゴリズム **BackTrack_Prune** の疑似コードを以下に示す．解が見つからなかったことを登録する機能を **EntPrune**，枝刈り判定を行う機能を **ChkPrune** として，それぞれ 1 行追加している．

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

以下，枝刈り条件の登録と判定をどのように実装するかについて述べる．

定理 4.4 の条件式は単純で美しいが，$n(Z), c(Z)$ の 2 次元のパラメータを扱うので，実装はそれほど簡単ではない．1 次元の不等式であれば最大値（ハイスコア）を憶えておくだけでよいが，2 次元の場合は極大（Pareto 最適）な解を記憶しておき，その中の少なくとも 1 つに対して条件式を満たすかどうかを判定する必要がある．しかし極大解の個数は指数関数的に多くなる可能性があり，記憶量や計算時間が膨大になってしまうおそれがある．

今回の枝刈りは無駄な計算を省くことが目的であり，判定を多少手抜きして枝刈りのチャンスを見逃したとしても計算の正しさには影響を与えない．枝刈りを見逃した場合でも次の再帰呼び出しでも必ず条件を満たすので，いずれは枝刈りが行われる．見逃しの確率があまり大きくなり過ぎると計算時間が著しく低下するが，完璧である必要はない．そこで我々は，図 2 に示す通り，$n(Z)$ と c(Z) の 2 次元空間を，まず $n(Z)$ の値により短冊状に k 分割 (k は適当に定める定数）し，それぞれの短冊について，過去に解が

見つからなかった最大の $c(Z)$ を記録しておくこととした．判定を行う際は，まず $n(Z)$ により短冊を選び，その領域で過去に解なしとなった $c(Z)$ の最大値を下回っていれば，解を見つけられる見込みはないので，再帰呼び出しを打ち切って枝刈りをしても計算結果に影響はない．

分割数 k をいくつにするかは実用上重要な問題である．k が大き過ぎるとメモリを浪費するだけでなく，ある短冊での $c(Z)$ の記録が更新されたとき，その短冊だけでなく近傍の短冊でも記録を更新できるかチェックしなければならないので計算時間も増大する．一方，k が小さすぎると，短冊の幅が粗くなり過ぎて，本来なら記録更新できた点が更新できなくなり，枝刈り検出の精度が低下する．最適な k の値は，データの規模や数値の分布に依存する．我々の実装では，今のところ実験的に k を決めており，区画数 m に対して，$k = 64 \times m$ としている．さらに $tail(Z)$ が同じもの同士で $n(Z), c(Z)$ を記録して比較する必要があるため，m 通りの異なる記録場所を別に用意しており，トータルで $64 \times m^2$ 個の整数値を記録するスペースを使用している．枝刈り判定の実装については，今後も検討を加える余地がある．

4.4 制約条件を追加した列挙

ある条件を満たすホットスポットだけを列挙したいという要望は実用上よくあることである．本アルゴリズムを若干書き換えるだけで，ある種の条件を満たすホットスポットだけに絞り込むことができる．

人口の上限を指定した解の列挙

人口 $n(Z)$ の上限 n_{max} を指定し，少人数のホットスポットだけを抽出するようにアルゴリズムを改造することは比較的容易である．探索中に再帰呼び出しをする度に領域の人口は増加する一方なので，どこかで n_{max} を超えたらその時点で再帰呼び出しを打ち切ればよい．また，過去の探索結果を参照して枝刈りが起こるときは，必ず $n(Z) \geq n(Z')$ となっているので，任意の $X \subseteq Z^r$ に対して $n(Z \cup X) \geq n(Z' \cup X)$ が成り立つ．したがって，もしも $n(Z' \cup X)$ が n_{max} を超えるため解にならないならば，$n(Z \cup X)$ も必ず n_{max} を超えるので解にならない．つまり，制約なしの場合と全く同様に枝刈りを行っても解を見逃すおそれはない．逆に，人口の下限を指定して大きなホットスポットだけを抽出しようとすると，枝刈りによって解を見逃すおそれが生じるので，そのままでは適用できない．

発生数の下限を指定した解の列挙

発生数 $c(Z)$ の下限 c_{min} を指定し，発生数が大きいホットスポットだけを抽出するようにアルゴリズムを改造することも比較的容易である．過去の探索結果を参照して枝刈りが起こるときは，必ず $c(Z) \leq c(Z')$ となっているので，任意の $X \subseteq Z^r$ に対して $c(Z \cup X) \leq c(Z' \cup X)$ が成り立つ．したがって，もしも $c(Z' \cup X)$ が c_{min} に達しないため解にならないならば，$c(Z \cup X)$ も決して c_{min} に達しないので解にならない．つまり，制約なしの場合と全く同様に枝刈りを行っても解を見逃すおそれはない．逆に，発生数の上限を指定して小さなホットスポットだけを抽出しようとすると，枝刈りによって解を見逃すおそれが生じるので，そのままでは適用できない．

以上の考察から，比較的小さいホットスポットに絞り込みたいなら人口で，大きいホットスポットに絞り込みたいなら発生数で制約をかけることにすれば，制約なしの場合とほとんど同様のアルゴリズムで効率よくホットスポットを列挙することができる．

ブロックサイズ上限を指定した解の列挙

ブロックサイズ（Z に含まれる区画数）の上限を指定し，小さなホットスポットだけを抽出するようにアルゴリズムを改造することも比較的容易である．本アルゴリズムは区画番号の辞書順に探索を行うので，まず人口の多い順にソートしてから区画番号を定め，指定された上限サイズを超えた場合にそれ以上の再帰呼び出しを行わないようにして探索を行えばよい．この方法でホットスポットを列挙すれば，枝刈りによって解を見逃すおそれがないことを保証できる．人口の多い順にソートして区画番号を定めて辞書順に探索したとすると，過去の探索結果を参照して枝刈りが起こるときは，必ず $n(Z) \geq n(Z')$ となっているが，このとき，辞書順で後から出現した Z では先に出現した Z' に比べて少ない人口の区画しか使えないので，人口で Z' を上回るためには，Z' と同数またはそれ以上の個数の区画を消費しているはずである．したがって今後追加できる残り区画数は Z' よりも多くなることはない．とすると，Z' 以下の再帰呼び出しで解なしであったとすると，Z 以下に新たな解が出現する見込みはない．したがって，制約なしの場合と全く同様に枝刈りを行っても解を見逃すおそれはない．

5. 実験と評価

以上で説明したアルゴリズムを実装し，実験と評価を行った．プログラムは C++で約 600 行，使用計算機は Let's Note CF-SZ6 (Intel Core i7 2.7GHz, メモリ 8GB)，使用 OS は Cygwin 2.10.0 on Windows 10 である．本プログラムは列挙した解をファイルに出力するモードと，解の個数のみを数えるモードを持つ．以下の実験では個数のみを数える場合の計算時間を示しており，ファイルに出力する場合はファイル書き込みのための時間を要する．なお，本プログラムでは，特に指定しない限りは，発生率が高い順に区画をソートしてから辞書順に探索を行っている．

本実験では，以下の 2 種類の実統計データを使用した．

- 全国 47 都道府県の自殺者数データ (H28 年度)[11]
- 米国 North Carolina 州 100 郡市の SIDS（乳幼児突然

表 2　47 都道府県データに対する実験結果

$\log\lambda$ 閾値	解の個数	探索回数	時間 (秒)
117.0	0	175,257	0.08
116.0	5	188,261	0.08
110.0	4,414	454,462	0.09
100.0	696,559	9,325,141	0.6
90.0	22,049,421	144,236,159	7.9
85.0	92,666,978	481,449,042	26.6
80.0	341,003,096	1,499,894,205	82.9

表 3　素朴な全探索法との性能比較

区画数	全探索法		提案手法	
	探索回数	時間 (秒)	探索回数	時間 (秒)
20	1,048,575	0.08	7,305	0.06
25	33,554,431	0.9	15,742	0.06
30	1,073,741,823	25.3	27,599	0.07
35	34,359,738,367	809.1	58,373	0.08
40	–	(time out)	101,273	0.08
45	–	(time out)	148,553	0.08
47	–	(time out)	175,257	0.08

表 4　100 区画 SIDS データに対する実験結果

$\log\lambda$ 閾値	解の個数	探索回数	時間 (秒)
68.0	0	4,155,197	0.3
67.7	2	4,170,643	0.3
67.5	41	4,209,677	0.3
67.0	1,582	5,250,495	0.4
66.5	19,850	11,095,802	0.7
66.0	152,525	35,967,019	2.1
65.5	901,043	126,295,120	7.1
65.0	4,437,311	421,469,840	23.7

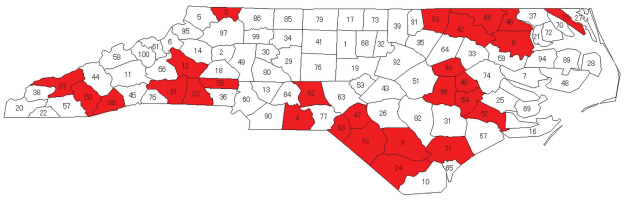
死症候群）発生数データ [3]

まず 47 都道府県データに対して，対数尤度比 $\log\lambda$ の閾値を変えながら実験した結果を表 2 に示す．実験結果から，ヒットする解の個数が少ないときは，極めて短時間で探索が終了していることがわかる．この探索と同時に $\log\lambda$ の最大値 116.341 を知ることができる．47 都道府県の規模のデータで，領域の形状や大きさに制限を加えることなしに，空間スキャン統計量の厳密な最大値が現実的な時間で求められたことはこれまでになく，特筆すべき結果である．

同じデータで，枝刈りを行わずに素朴に全探索を行う方法と比較した結果を表 3 に示す．この実験では，十分大きな閾値を与えて探索結果は解なしとして，探索中に得られる $\log\lambda$ 最大値のみを求めている．全探索法では，47 都道府県をすべて使うと時間がかかり過ぎるため，都道府県番号が若い方から k 個の区画だけを取り出した例題で比較を行った．この実験結果から，枝刈りによる顕著な高速化効果が読み取れる．全探索では 47 都道府県を探索するためには計算上 40 日程度かかると推定され，この例題においては 100 万倍を超える高速化効果が得られている．

次に，100 区画からなる SIDS データで実験を行った結果を表 4 に示す．このデータはホットスポット探索の分野で良く使われてきた有名な例題であるが，これまでに，全

対数尤度比 1 位 ($\log\lambda = 67.720$)



対数尤度比 2 位 ($\log\lambda = 67.711$)

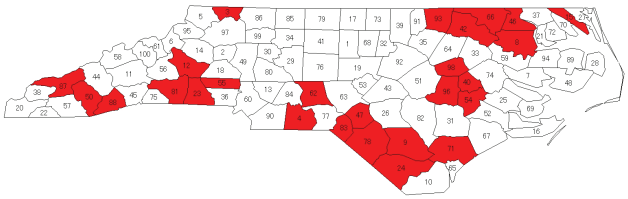


図 3　SIDS データに対する上位 2 件の結果

表 5　最大 200 区画までの性能比較

区画数	$\log\lambda$ 最大値	探索回数	時間 (秒)
100	67.720	4,155,197	0.3
110	76.755	6,587,933	0.5
120	76.966	12,085,785	0.8
130	84.852	25,797,420	1.6
140	89.186	61,264,514	3.7
150	102.870	92,613,939	5.8
160	107.693	200,752,599	11.6
170	112.732	287,506,588	18.9
180	119.536	432,613,670	24.7
190	128.952	515,692,339	31.9
200	135.439	742,498,024	42.2

てのべき集合に関する空間スキャン統計量の最大値を求めたという報告はなく，世界で初めて明らかになった結果である．図 3 に上位 2 件の解を示す．この結果から，統計量が最大となる解には飛び地が多く含まれていることがわかる．

本手法の限界を調べるため，上記の SIDS データを 2 つ複製してマージすることで，人為的に 200 区画のデータを作成し，その中から番号の若い k 個の区画を選び入力データとした．そして十分に大きい閾値を与えて $\log\lambda$ 最大値のみを求める実験を行った．結果を表 5 に示す．この実験結果によれば，緩やかではあるが指数関数的に計算時間が伸びていることが観察できる．本アルゴリズムは，枝刈りが起きるまでは二分木の探索になるので，最初に枝刈りが発生するまでの平均の探索深さを d とすると，解なしの場合でも，$O(2^d)$ 程度の計算時間はかかると考えられる．より精密な計算量の解析は今後の検討課題である．

最後に，100 区画からなる SIDS データでブロックサイズの上限を指定して，十分に大きい閾値を与えて $\log\lambda$ 最大値のみを求める実験を行った．実験結果を表 6 に示す．上限サイズが小さいほど高速に列挙できることが観察できる．この例題で $\log\lambda$ が最大となる解は 28 区画からなるため，28 より大きな上限サイズを指定しても出力結果は

表 6　ブロックサイズ上限を指定した探索結果

上限サイズ	$\log\lambda$ 最大値	探索回数	時間 (秒)
2	15.969	5,050	0.06
3	23.635	60,084	0.08
5	36.792	723,436	0.1
10	49.229	11,601,667	0.7
15	59.342	55,023,759	3.3
20	65.900	150,175,694	8.0
25	67.646	329,125,624	17.6
30	67.720	657,855,752	35.4
99	67.720	2,736,200,411	152.1
−	67.720	4,155,197	0.3

変わらない．ただし計算時間は上限サイズに依存する．ブロックサイズ上限を指定する場合は，人口の多い順に区画をソートしてから辞書順に探索する必要があるが，この変数順序は枝刈りが効きにくいことが観察できる．表の最下段はブロックサイズ上限を指定しない場合に，発生数の多い順にソートした場合の結果を示している．こちらの方が 100 倍以上速いことがわかる．

6. おわりに

本研究では，空間スキャン統計量に部分的な単調性があることを見出し，その性質を利用した探索空間の枝刈りを用いて非常に高速なホットスポット抽出列挙アルゴリズムを実現した．実験の結果，全国 47 都道府県の例題に対して，素朴な全探索法に比べて 100 万倍以上高速に，統計量最大のホットスポットを抽出することが可能となった．本手法はおよそ 100〜200 区画程度までの統計データに対して現実的な計算時間で動作する．

本手法は，統計量が閾値を上回るようなホットスポットを全て抽出し列挙するが，解の個数が多くなり過ぎると人間が見ても理解できないので，表示方法にも工夫が必要である．例えば，各区画が解に含まれる頻度を数えてヒートマップとして表示するのも有力な手段と思われる．

本手法は，飛び地を許した非連結な領域も含むすべてのべき集合を探索対象としている．応用の場面では，連結な解だけが欲しいという場合もあると思われるが，非連結解が大量にヒットして連結解がなかなか見つからないケースも考えられる．連結解だけを効率よく探索するためには，BDD/ZDD（二分決定グラフ）を用いた技法 [5] を取り入れ，連結の条件を BDD で表現しながら，ホットスポットを探索する方法の研究開発が必要と思われる．

本研究では，ホットスポットの有意性検定の実験はまだ行っていないが，従来の方法ではモンテカルロ法により 999 通りのランダムデータを発生させて探索を繰り返す必要がある．本手法により 1 回の探索時間が大幅に短縮されているので，検定にかかる時間の短縮効果も非常に大きいと期待される．さらに今後の課題として，モンテカルロ法によらず解析的に多重検定の p 値を求める手法も検討したい．

何らかの基準で多数の区画に区分して母集団の数と属性を持つものの数（発生数）を記録した統計データは，世の中に大量に存在している．これを任意の 2 群に分けたときの偏りを解析することは極めて基本的な操作であり，応用は幅広い．生命科学，疫学，商業，農業，工業，気象，交通，教育など様々な分野への応用が期待される．

謝辞　本研究を進めるにあたり，予備的実験と考察に取り組んだ石丸亮さん，および議論して頂いた北大情報科学研究科 大規模知識研究室の皆様に感謝いたします．本研究の一部は JSPS 科研費基盤 (S) 15H05711 の助成による．

参考文献

[1] Anselin, L.: Local indicators of spatial association-LISA, *Geographic Analysis*, Vol. 27, No. 2, pp. 93–115 (1995).
[2] Besag, J. and Newell, J.: The Detection of Clusters in Rare Diseases, *Journal of the Royal Statistical Society, Series A*, Vol. 154, No. 1, pp. 143–155 (1991).
[3] Cressie, N. A. and Chan, N.: Spatial modeling of regional variables, *Journal of The American Statistical Association*, Vol. 84, No. 406, pp. 393–401 (1989).
[4] Ishioka, F., Kurihara, K., Suito, H., Horikawa, Y. and Ono, Y.: Detection of hotspots for three-dimensional spatial data and its application to environmental pollution, *Journal of Environmental Science for Sustainable Society*, Vol. 1, pp. 15–24 (online), DOI: 10.3107/jesss.1.15 (2007).
[5] Kawahara, J., Horiyama, T., Hotta, K. and Minato, S.: Generating All Patterns of Graph Partitions within a Disparity Bound, *Proc. of the 11th International Workshop of Algorithms and Computation (WALCOM2017), (LNCS 10167, Springer)*, pp. 119–131 (2017).
[6] Kulldorff, M.: A spatial scan statistic, *Communications in Statistics - Theory and Methods*, Vol. 26, No. 6, pp. 1481–1496 (online), DOI: 10.1080/03610929708831995 (1997).
[7] Openshaw, S., Charlton, M., Wymer, C. and Craft, A.: A Mark 1 Geographical Analysis Machine for the automated analysis of point data sets, *International Journal of Geographical Information Systems*, Vol. 1, No. 4, pp. 335–358 (online), DOI: 10.1080/02693798708927821 (1987).
[8] Tango, T. and Takahashi, K.: A flexibly shaped spatial scan statistic for detecting clusters, *International Journal of Health Geographics*, Vol. 4, No. 11 (2005).
[9] Uno, T., Kiyomi, M. and Arimura, H.: LCM ver.2: efficient mining algorithms for frequent/closed/maximal itemsets, *Proc. IEEE ICDM'04 Workshop FIMI'04 (International Conference on Data Mining, Frequent Itemset Mining Implementations)* (2004).
[10] 栗原考次：階層的空間構造を利用したホットスポット検出，計算機統計学， Vol. 15, No. 2, pp. 171–183 (2002).
[11] 警察庁：平成 28 年中における自殺の状況, https://www.npa.go.jp/publications/statistics/safetylife/ (2016).
[12] 石丸　亮：統計データのホットスポットの ZDD 表現に関する研究, 北海道大学大学院情報科学研究科 修士論文 (2018).