Maximizing Multi-scale Spatial Statistical Discrepancy

Weishan Dong[†], Renjie Yao[†‡], Chunyang Ma[†], Changsheng Li[†],
Lei Shi[♯*], Lu Wang[§], Yu Wang[†], Peng Gao[†], Junchi Yan[†]
[†]IBM Research – China
[‡]Northeastern University, China
[♯]SKLCS, Institute of Software, Chinese Academy of Sciences
[§]University of Chinese Academy of Sciences
{dongweis,machybj,lcsheng,yuwangbj,bjgaop,yanjc}@cn.ibm.com,
yrjyrjwp7@hotmail.com, shil@ios.ac.cn, luwang@ucas.ac.cn

ABSTRACT

Detecting anomalous events from spatial data has important applications in real world. The spatial scan statistic methods are popular in this area. With maximizing the spatial statistical discrepancy by comparing observed data with a given baseline data distribution, significant spatial *overdensity* and *underdensity* can be detected. In reality, the spatial discrepancy is often irregularly shaped and has a structure of multiple spatial scales. However, a large-scale discrepancy pattern may not be significant when conducting fine granularity analysis. Meanwhile, local irregular boundaries of a maximized discrepancy cannot be well approximated with a coarse granularity analysis. Existing methods mostly work either on a fixed granularity, or with a regularly shaped scanning window. Thus, they have difficulties in characterizing such flexible spatial discrepancies. To solve the problem, in this paper we propose a novel discrepancy maximization algorithm, `RefineScan`. A grid hierarchy encoding multi-scale information is employed, making the algorithm capable of maximizing spatial discrepancies with multi-scale structures and irregular shapes. Experiments on a wide range of datasets demonstrate the advantages of `RefineScan` over the state-of-the-art algorithms: It always finds the largest discrepancy scores and remarkably better characterizes multi-scale discrepancy boundaries. Theoretical and empirical analyses also show that `RefineScan` has a moderate computational complexity and a good scalability.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data Mining*

Keywords

Anomalous event detection; spatial scan statistic; multi-scale statistical discrepancy

[*]The work of Lei Shi is supported by China National 973 project 2014CB340301 and NSFC grant 61379088.

CIKM'14, November 3–7, 2014, Shanghai, China.

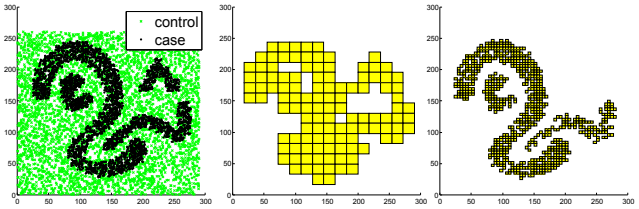
http://dx.doi.org/10.1145/2661829.2662007.

1. INTRODUCTION

Anomalous event detection is an important problem in data mining. It has numerous applications in environmental monitoring, epidemic surveillance, criminology research, etc. The spatial scan statistic [9] is one of the most popular methods in detecting anomalous events from spatial datasets, where the events are geo-referenced. In such a context, the anomaly is defined as *the unusually high (or low) density of events in a geographically bounded region that is unlikely to have occurred only by chance.* To quantify such anomalies, a statistical discrepancy score between observed data and a given baseline distribution is evaluated and maximized. A p-value that reflects the unusualness of the maximized discrepancy is also computed by hypothesis testing. Such an analysis is also referred to as *overdensity/underdensity* detection [1, 5] or *cluster* detection [7, 12] where the region maximizing the discrepancy score is called a cluster[1]. Take the epidemic surveillance scenario as an example, the people infected by an epidemic disease in a city are observed data, which are called *cases*, and the people not infected are called *controls*. The summation of cases and controls is called *population*, which defines the baseline distribution. Given the locations of each data sample, maximizing the spatial statistical discrepancy between the cases and the population can identify city regions with elevated disease risk (overdensity). Suppose there is a high density of population in the city center, which naturally leads to a high density of disease cases observed there if the probability of getting infected is uniform throughout the population. However, such a clustering of cases caused by the inhomogeneous baseline distribution is not anomalous. Only the *extraordinary* clustering effect that exceeds the interpretability of the baseline is anomalous, which implies a higher probability of getting infected than expected. The discrepancy maximization is well suited for such anomaly detection applications.

By searching over the geographical space, regions maximizing a discrepancy function (i.e., clusters) can be identified. In reality, the discrepancy can often be highly flexible in its geospatial structure. On one hand, it may embed com-

[1]The difference between the cluster concept here and the one in the classical clustering problem is significant. In the classical clustering problem, the focus is the overall data clustering effect instead of the statistical discrepancy defined between observations and a baseline. Therefore, in the classical clustering framework, the statistical significance is usually not considered, and detecting underdensity is impossible. More discussions can also be found in [1, 5].



(a) Dataset1 (b) 16×16 grids (c) 64×64 grids

Figure 1: A sample dataset and overdensity regions (significant at 0.05 level) detected by GridScan [5]

plex multi-scale structures that cannot be well described at one spatial scale only. Especially, maximized discrepancies obtained at different scales can be far different, which we will discuss later. On the other hand, the shape of the cluster can be irregular at different scales. We refer to it as the *multi-scale spatial discrepancy*, which is ubiquitous in spatial data. Figure 1(a) shows a typical example: a uniform distribution of controls is overlaid with four overdensity regions of cases that are of varied scales. The capability of modeling such spatial discrepancies can be critical, since a more accurate characterization of anomalies undoubtedly helps achieve stronger situational awareness in real-world applications [17, 10, 2, 8, 5]. However, existing scan statistic based algorithms mostly work on simplified data assumptions and often fail when dealing with multi-scale spatial discrepancies. Mainly due to computational reasons, earlier proposed algorithms assume that the clusters have regular shapes, e.g., circle [9], ellipse [10], and rectangle [12, 1]. Although the detected clusters can be of varied scales, these algorithms easily fail to detect irregularly shaped clusters, making their applications limited in practice. Recently proposed algorithms can detect irregularly shaped clusters [17, 2, 8, 4, 5], but most of which, if not all, work at a fixed spatial scale. Their common assumption is that the input data are geographically aggregated by areas with fixed boundaries. Sizes and shapes of the areas are predefined by either administrative regions or regular grids. Because the spatial granularity is fixed, the spatial discrepancies at finer or coarser granularities cannot be addressed, which is well-known in geographical analysis as the modifiable areal unit problem (MAUP) [15]. In other words, these algorithms solve the single-scale discrepancy maximization problem. Figure 1 demonstrates examples of applying a typical grid-based algorithm, GridScan [5], with two different grid sizes on the sample dataset. We can see that when using 16×16 grids (relatively large grid units), a large-scale cluster significant at 0.05 level is detected, and it connects all the local overdensity regions. But apparently, the grids are too coarse to approximate the local shapes. When using 64×64 grids (relatively small grid units), the fine-grained overdensity is better characterized but the global shape also becomes fragmented. In addition, because the cluster locating in the upper right of Figure 1(a) disconnects with the others at the given scale, its discrepancy score is too small to be identified as significant (at 0.05 level). The limitation of analyzing a single scale is evident.

Nonetheless, determining an optimal spatial scale for a geographical analysis is difficult in general. In fact, according to the MAUP, the possibly different (and sometimes even contradictory) spatial analysis results obtained at different scales are all correct [15]. Thus, the superiority of any single scale over the others is usually hard to measure. However, specifically in the cluster detection problem, the maximized discrepancy scores obtained at different scales are numerically comparable. We argue that rather than striving to find an "optimal" scale, a better strategy dealing with multi-scale discrepancies can be: combining the information gathered from different scales to search for the region globally maximizing the discrepancy score. To the best of our knowledge, no algorithm so far can incorporate multi-scale information for spatial discrepancy maximization. Compared with the single-scale analyses, there are three technical challenges in maximizing a multi-scale spatial discrepancy: (1) Determining boundaries of the discrepancy is much harder than working on a fixed areal aggregation. Because there is no available reference boundary at larger or smaller scales, the search space is tremendously enlarged. (2) Due to the MAUP, cluster detection results can be far different at different scales when the discrepancy indeed embeds multi-scale structures. A reasonable way of combining such information has to be developed. (3) Algorithms meeting the two aforementioned challenges will inevitably introduce additional computations. The overall computational complexity must be acceptable in practice.

To meet these challenges, in this paper, we propose a novel spatial discrepancy maximization algorithm, `RefineScan`. The main idea is as follows. (1) On top of a base level spatial aggregation (by regular grids in this paper) that represents the finest granularity in which the user is interested, a multi-level grid hierarchy is built for information diffusion across multiple scales. (2) On a higher level of the hierarchy, the grid unit size becomes larger. Region search is initially done on the top level grids to locate large-scale, yet coarse, clusters. (3) The coarse clusters are then iteratively refined on lower levels until reaching the base level, so that the useful information for maximizing the discrepancy is kept and passed on to lower levels. An additional technical challenge is that, due to the mathematical characteristics of the employed discrepancy function (i.e., the scan statistic likelihood), there is theoretical possibility of cluster merging and splitting during refinement. In-depth analyses are given from this perspective, and efficient computational methods are developed accordingly. By doing so, the clusters detected by `RefineScan` not only preserve the large scale spatial connectedness that may help globally maximize the discrepancy, but also accurately characterize small scale irregular shapes. Experiments will show that compared with existing algorithms, `RefineScan` performs the best in maximizing multi-scale spatial discrepancy and characterizing the flexible clusters given ground truths. The computational complexity of `RefineScan` is also moderate: $O(n + N^3)$ given $N \times N$ base level grids and a dataset of size n.

The remainder of the paper is organized as follows. Section 2 details `RefineScan` and analyzes its computational complexity. Section 3 presents experimental studies. Section 4 reviews related work. Section 5 concludes the paper.

2. PROPOSED APPROACH

Table 1 summarizes the notations used in this paper. Kulldorff's scan statistic [9] is one of the most popular discrepancy measures developed in literature. Given a region Z, the likelihood function of the scan statistic, $L(Z)$, is defined as the objective function of discrepancy maximization. Two widely used probabilistic models for defining $L(Z)$ are Bernoulli and Poisson. Formally, when data are typical binary events, the Bernoulli model is used. Then we have

Table 1: Notations

Symbol	Description
D	input spatial dataset
n	number of population in D, $n = \|D\|$
m	number of cases in D, $m \leq n$
$N \times N$	dimension of base level regular grids
Z	a cluster (region) defined by connected cells
n_Z	number of population inside Z
m_Z	number of cases inside Z
l_{max}	number of levels of grid hierarchy, $\log_2 N \geq l_{max} \geq 1$
l	level indicator of grid hierarchy, $l_{max} \geq l \geq 0$
$\mathcal{C}^{(l)}$	set of all grid cells on level l, $\|\mathcal{C}^{(l)}\| = \frac{N}{2^l} \times \frac{N}{2^l}$
c	a grid cell on a given level, $c \in \mathcal{C}^{(l)}$
n_c	number of population inside cell c
m_c	number of cases inside cell c
$\mathcal{U}$	set of unmarked cells on a given level, $\mathcal{U} \subseteq \mathcal{C}^{(l)}$
K	number of Monte Carlo simulations

$$L(Z) = \left(\frac{m_Z}{n_Z}\right)^{m_Z} \times \left(1 - \frac{m_Z}{n_Z}\right)^{n_Z - m_Z} \times \left(\frac{m - m_Z}{n - n_Z}\right)^{m - m_Z} \times \left(1 - \frac{m - m_Z}{n - n_Z}\right)^{(n - n_Z) - (m - m_Z)} \quad (1)$$

When the number of cases in data is known as Poisson distributed, the Poisson model is used, and

$$L(Z) = \left(\frac{m_Z}{E(m_Z)}\right)^{m_Z} \times \left(\frac{m - m_Z}{m - E(m_Z)}\right)^{m - m_Z} \quad (2)$$

where $E(m_Z) = \frac{m}{n} \times n_Z$ is the expected number of cases in a region under the null hypothesis of no overdensity or underdensity. Also note that additional baseline information other than locations can be considered here. For instance, variables such as age and sex in the epidemic surveillance application mentioned in Section 1 can be treated as *covariates*, and $E(m_Z)$ becomes the *covariate adjusted* expectation [9, 11]. A cluster is found by $\arg\max L(Z)$. Local optima of $L(Z)$ also need to be found, because multiple local clusters can exist, which are necessary to be detected in practice. When detecting overdensity, a constraint $\frac{m_Z}{n_Z} > \frac{m - m_Z}{n - n_Z}$ is applied, meaning only regions with more cases observed than expected are to be detected. Conversely, $\frac{m_Z}{n_Z} < \frac{m - m_Z}{n - n_Z}$ is applied for underdensity detection. Because the distribution of $L(Z)$ is unknown, Monte Carlo simulation is usually adopted to calculate the p-value when evaluating the significance of a cluster [9]. Specifically, a number of replicated datasets are generated by randomly reassigning the case and control labels under the null hypothesis. On each replicated dataset, $L(Z)$ is maximized, which refers to one simulation. Suppose a cluster Z' has a discrepancy score $L(Z')$, then its p-value is $p = \frac{r+1}{K+1}$, where K is the number of simulations, and r is the number of discrepancy scores obtained from the simulations with values no smaller than $L(Z')$. In implementations, $\log L(Z)$ is often used instead of $L(Z)$, so we will use $\log L(Z)$ in algorithm descriptions hereafter. Algorithm 1 outlines the overall cluster detection framework. Line 1 is detecting all clusters by `RefineScan`. Lines 2–6 are Monte Carlo simulations. Lines 7–9 are computations of p-values. Function $I(\cdot)$ on line 8 is the indicator function.

Algorithm 1: ClusterDetection

```
   Input: D; K
   Output: A set of clusters, clusters
1  clusters ← RefineScan(D);
2  KClusters ← ∅;
3  for i ← 1 to K do                          // Monte Carlo simulations
4      generate a replicated dataset D^i;
5      tmpClusters ← RefineScan(D);
6      KClusters ← KClusters ∪ {arg max_{Z' ∈ tmpClusters} log L(Z')};
7  foreach Z ∈clusters do                     // compute p-values
8      r ← Σ_{Z' ∈ KClusters} I(log L(Z') ≥ log L(Z));
9      p_Z ← (r+1)/(K+1);
10 return clusters;
```

2.1 RefineScan Algorithm

To handle the multi-scale spatial discrepancy, a grid hierarchy data structure is employed in `RefineScan`. Suppose $N \times N$ grids are applied to the data in D as the base level grids. A base level grid cell is either marked or unmarked, indicating its state of whether being a part of a cluster or not. We define that a higher level regular grid cell is the parent of four lower level grid cells, and a lower level cell

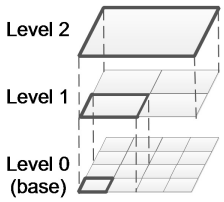


Figure 2: An example of three-level grid hierarchy

equals one quadrant of a parent cell. In this way, the base level $N \times N$ grids correspond to $\frac{N}{2} \times \frac{N}{2}$ grids on a higher level, namely level one, and further build up $\frac{N}{2^2} \times \frac{N}{2^2}$ grids on level two, and so on. On level l, there are $\frac{N}{2^l} \times \frac{N}{2^l}$ grids. Figure 2 shows a simple example of a three-level grid hierarchy. On a given level of the grid hierarchy, a cluster is a set of connected cells. In this paper, we consider the immediate neighboring relationship defined by the 8-connectedness (i.e., the first-order Queen spatial contiguity). We regard cells inside a cluster as *marked* and the rest *unmarked*. Let $\mathcal{U}$ denote all the unmarked cells on a level. `Get8NB`(c) is a function returning all the 8-connected neighboring cells of cell c. The inner-neighbors and outer-neighbors of a cluster are defined as follows.

Definition 1 (Inner-Neighbors). *Inner-neighbors (*inNB*) of a cluster Z are the marked cells of Z neighboring $\mathcal{U}$:* $\mathsf{inNB} = \{c | c \in Z, \texttt{Get8NB}(c) \cap \mathcal{U} \neq \emptyset\}$.

Definition 2 (Outer-Neighbors). *Outer-neighbors (*ouNB*) of a cluster Z are the unmarked neighboring cells of Z:* $\mathsf{ouNB} = \{c | c \in \mathcal{U}, \texttt{Get8NB}(c) \cap Z \neq \emptyset\}$.

We use `GetInNB`(Z) and `GetOuNB`(Z) to denote the functions returning all inner- and outer-neighbors of Z, respectively.

The flow of `RefineScan` is shown in Algorithm 2, which is a two-step procedure. After initializing the grid hierarchy (lines 1–3), in the first step (lines 4–8), the rough locations of candidate clusters at the largest available scale are identified by region-growing on the top level grids. The procedure of locating a large-scale cluster is `LocateCluster` (Algorithm 3). In the second step (lines 9–15), each of the large-scale and coarse clusters is refined on lower level grids, such that the spatial structure at a finer level is exploited. The refinement is done on each level of the hierarchy in a top-down manner. Before refining on a lower level, all clusters and $\mathcal{U}$ are reconstructed by decomposing every component grid unit into its four children grids on the lower level. Then, each cluster is iteratively refined by either removing cells from the inner-neighbors, or growing cells from the outer-neighbors. Once the refinement of all clusters finishes on one level, the same operations are repeated on the next available lower level. Apparently, the granularity of a cluster becomes finer as

Algorithm 2: RefineScan

Input: $D; N; l_{max}$; minP.
Output: A set of clusters, clusters.

define $\mathcal{C}^{(0)}$ with $N \times N$ regular base level grids;
foreach $c \in \mathcal{C}^{(0)}$ **do** set m_c and n_c based on D;
build grid hierarchy $H = \{\mathcal{C}^{(0)}, \mathcal{C}^{(1)}, \ldots, \mathcal{C}^{(l_{max})}\}$;
// locate large-scale clusters on top level grids
$\mathcal{U} \leftarrow \mathcal{C}^{(l_{max})}$; clusters $\leftarrow \emptyset$; candSeedSet $\leftarrow \mathcal{U}$;
while candSeedSet $\neq \emptyset$ **do**
  {Z, candSeedSet, $\mathcal{U}$, clusters} $\leftarrow$ LocateCluster(candSeedSet, $\mathcal{U}$, clusters, minP);
  if $Z \neq \emptyset$ **then** clusters $\leftarrow$ clusters $\cup \{Z\}$;
  else break;
// refine clusters to characterize local shapes
$l \leftarrow l_{max} - 1$;
while $l \geq 0$ **do**
  decompose grids in each $Z \in$ clusters and $\mathcal{U}$ to $\mathcal{C}^{(l)}$ grids;
  label each $Z \in$ clusters as *unrefined*;
  foreach unrefined $Z \in$ clusters **do**
    {$\mathcal{U}$, clusters} $\leftarrow$ RefineCluster($Z, \mathcal{U}$, clusters, minP);
  $l \leftarrow l - 1$;
return clusters;

the level goes down. Meanwhile, because the principle of refinement is to monotonically enlarge the discrepancy scores, the grids' spatial connectedness at larger scales is retained also at smaller scales as long as it is helpful for enlarging the score. It can be seen as information diffusions from higher to lower grid levels. In this way, the information of different scales is combined for cluster detection. The procedure of refining a cluster is `RefineCluster` (Algorithm 5).

2.1.1 LocateCluster (Algorithm 3)

The objective here is to roughly and also quickly locate a cluster on the top level grids. It can be inferred that an exhaustive search needs to test $2^{|\mathcal{C}^{(l_{max})}|} = 2^{(\frac{N}{2^{l_{max}}})^2}$ alternatives to find the optimal solution, which is computationally inefficient. Therefore, here we adopt a greedy yet effective cluster growing strategy proposed in GridScan [5], which only has a linear time complexity to $|\mathcal{C}^{(l_{max})}|$. Briefly, it first finds a seed cell for a cluster, and then locally grows the cluster by its outer-neighbors. The seed is chosen from a candidate seed set (candSeedSet) by maximizing $\log L(\{c\})$. When growing a cluster, only one cell chosen from the outer-neighbors is included (and marked) each time. The cell is found by maximizing a discrepancy gain function $G_g(Z, c)$.

We first define a function $S_g(Z, c)$ before formulating $G_g(Z, c)$. The aforementioned cluster growing is greedy in nature. Therefore, it is possible that a growing cluster Z can approach one or more (at most three under 8-connectedness) existing clusters that have stopped growing in previous iterations. Let mergClusters $= \{Z' | c \in \texttt{GetOuNB}(Z'), Z' \in$ clusters$\}$ denote such clusters, where set clusters denotes existing clusters. If there exists a cell $c \in \texttt{GetOuNB}(Z) \cap \texttt{GetOuNB}(Z')$, then marking c will connect Z and Z'. In this case, the supposed discrepancy score of Z by growing c is:

$$S_g(Z, c) = \log L(Z \cup \{c\} \cup \text{mergClusters}) \tag{3}$$

It is known that $S_g(Z, c)$ is not surely greater than both $\log L(Z)$ and $\log L(Z')$ [5]. If $\exists Z' \in$ mergClusters, s.t. $S_g(Z, c) < \log L(Z')$, c will not be chosen for growing. This is to restrict the discrepancy score to monotonically increase. It can be inferred that merging Z with Z' by c to obtain a larger size cluster $Z'' = Z \cup \{c\} \cup Z'$ requires the following two conditions: (1) $L(Z') > L(Z' \cup \{c\})$, indicating that Z' did not grow by c in earlier search, and (2) $L(Z'') > L(Z') \bigwedge L(Z'') > L(Z)$, indicating that Z'' has a larger discrepancy score than both Z and Z'. It is easy to prove that this never happens when $L(Z)$ is monotonic with $\frac{m_Z}{n_Z}$. This is because when finding overdensity, if $\frac{m_{Z'}}{n_{Z'}} > \frac{m_{Z'}+m_c}{n_{Z'}+n_c}$, we have either $\frac{m_{Z'}+m_Z+m_c}{n_{Z'}+n_Z+n_c} < \frac{m_{Z'}}{n_{Z'}}$ or $\frac{m_{Z'}+m_Z+m_c}{n_{Z'}+n_Z+n_c} < \frac{m_Z}{n_Z}$. When finding underdensity, such a conclusion also holds with reversing the directions of the inequations. However, because $L(Z)$ is neither monotonic with $\frac{m_Z}{n_Z}$ nor with n_Z, cluster merging is likely to happen. The discrepancy gain function of growing a cell, $G_g(Z, c)$, is:

$$G_g(Z, c) = \begin{cases} 0 & \text{if } \exists Z' \in \text{mergClusters} \\ & \text{s.t. } S_g(Z, c) < \log L(Z') \\ S_g(Z, c) - \log L(Z) & \text{otherwise} \end{cases} \tag{4}$$

When choosing the seed or cell to grow, an additional condition is `AboveMinP` (Algorithm 4). It is to restrict extra large size of a cluster and make the cluster shape compact. A parameter minP is applied to limit the inclusion of cells that contain too less points. Empirical studies have shown that a default minP =1 works well enough in many cases [5].

2.1.2 RefineCluster (Algorithm 5)

Given that the clusters and $\mathcal{U}$ are decomposed to the current level grids, refining a cluster Z is also iterative and incremental. In each step, two operations are evaluated: (1) removing a cell from Z's inner-neighbors, or (2) growing a cell from Z's outer-neighbors. The one better enlarges Z's discrepancy score will be applied. Because the discrepancy

Algorithm 3: LocateCluster

Input: Candidate seeds, candSeedSet; $\mathcal{U}$; clusters; minP.
Output: New cluster Z; Updated candSeedSet; Updated $\mathcal{U}$; Updated clusters.

$Z \leftarrow \emptyset$; ouNB $\leftarrow \emptyset$;
$c_0 \leftarrow \arg\max_{c \in \text{candSeedSet}} \log L(\{c\})$; // find seed cell
if $\log L(\{c_0\}) > 0$ **and** AboveMinP (c_0, minP) **then**
  $\mathcal{U} \leftarrow \mathcal{U} \setminus \{c_0\}$; // mark seed cell
  $Z \leftarrow \{c_0\}$; // initialize Z
  ouNB $\leftarrow$ Get8NB(c_0);
  while *true* **do**
    $c_{next} \leftarrow \arg\max_{c \in \text{ouNB}} G_g(Z, c)$;
    if $G_g(Z, c_{next}) > 0$ **and** AboveMinP(c_{next}, minP) **then**
      mergClusters $\leftarrow \{Z' | c_{next} \in \texttt{GetOuNB}(Z'), Z' \in$ clusters$\}$;
      foreach $Z' \in$ mergClusters **do**
        $Z \leftarrow Z \cup Z'$; // merge with Z
        ouNB $\leftarrow$ ouNB $\cup$ GetOuNB(Z');
        clusters $\leftarrow$ clusters $\setminus \{Z'\}$;
      $\mathcal{U} \leftarrow \mathcal{U} \setminus \{c_{next}\}$; // mark cell c_{next}
      $Z \leftarrow Z \cup \{c_{next}\}$; // grow Z by c_{next}
      ouNB $\leftarrow$ ouNB $\setminus \{c_{next}\} \cup$ (Get8NB(c_{next}) $\cap \mathcal{U}$);
    else
      candSeedSet $\leftarrow$ candSeedSet $\setminus (Z \cup$ ouNB);
      break; // stop growing
return {Z, candSeedSet, $\mathcal{U}$, clusters};

Algorithm 4: AboveMinP

Input: Grid cell c; minP.
Output: A boolean value, *true* or *false*.

switch *cluster detection objective* **do**
  case *overdensity* **return** $(m_c \geq \text{minP})$;
  case *underdensity* **return** $(n_c - m_c \geq \text{minP})$;

Algorithm 5: RefineCluster

Input: Unrefined cluster Z; $\mathcal{U}$; clusters; minP.
Output: Updated $\mathcal{U}$; Updated clusters.

```
1  inNB ← GetInNB(Z);
2  ouNB ← GetOuNB(Z);
3  while true do
4      c_in ← arg max_{c∈inNB} G_r(Z, c);
5      c_out ← arg max_{c∈ouNB} G_g(Z, c);
6      if G_r(Z, c_in) = G_g(Z, c_out) = 0 then
7          label Z as refined;                          // stop refining
8          break;
9      if G_r(Z, c_in) ≥ G_g(Z, c_out) then
10         spltClusters ← {Z' | ⋂Z' = ∅, Z = ⋃Z' ∪ {c_in}};
11         foreach Z' ∈ spltClusters do
12             Z ← Z \ Z';                              // split new cluster
13             label Z' as unrefined;
14             clusters ← clusters ∪ {Z'};
15         U ← U ∪ {c_in}; Z ← Z \ {c_in};               // remove c_in
16         nb ← Get8NB(c_in);
17         inNB ← inNB \ {c_in} ∪ (nb ∩ Z);
18         ouNB ← ouNB∪{c_in}\{c|c ∈ nb∩U, c ∉ GetOuNB(nb∩Z)};
19     else if AboveMinP(c_out, minP) then
20         mergClusters ← {Z'|c_out ∈ GetOuNB(Z'), Z' ∈ clusters};
21         foreach Z' ∈ mergClusters do
22             Z ← Z ∪ Z';                              // merge existing cluster
23             inNB ← inNB ∪ GetInNB(Z');
24             ouNB ← ouNB ∪ GetOuNB(Z');
25             clusters ← clusters \ {Z'};
26         U ← U \ {c_out}; Z ← Z ∪ {c_out};              // grow c_out
27         nb ← Get8NB(c_out);
28         if nb ∩ U = ∅ then
29             inNB ← inNB \ nb;                        // fill special case hole
30         else
31             inNB ← inNB ∪ {c_out} \ {c|c ∈ nb ∩ Z, c ∉
               GetOuNB(nb ∩ U)};
32         ouNB ← ouNB \ {c_out} ∪ (nb ∩ U);
33 return {U, clusters};
```

score is monotonically increased, if a large-scale cluster has been roughly located on a higher level, its component grids' spatial connectedness can still be preserved on lower levels as long as its discrepancy score keeps increasing.

Removing a cell from inner-neighbors is implemented on lines 9–18. This may cause a cluster to split into multiple (at most four under 8-connectedness) pieces. Let $\mathsf{spltClusters} = \{Z' | \bigcap Z' = \emptyset, Z = \bigcup Z' \cup \{c\}\}$ denote the new clusters that will split out of Z once removing c. Z will then become $Z'' = Z \setminus \{c\} \setminus \mathsf{spltClusters}$. The supposed maximized discrepancy score generated by removing c from Z is:

$$S_r(Z, c) = \max_{Z' \in \mathsf{spltClusters} \cup \{Z''\}} \log L(Z') \tag{5}$$

Similar to the analysis in Section 2.1.1, $S_r(Z, c)$ is not always greater than $\log L(Z)$ or $\log L(Z')$, where $Z' \in \mathsf{spltClusters} \cup \{Z''\}$. Therefore, cluster splitting is probable as well. When judging if removing c can result in cluster splitting, just considering c and its 8-neighbor is necessary but not sufficient. There are $2^8 - 1 = 255$ possible situations of the 8-neighbor. The excluded situation refers to that the 8-neighbor cells are all unmarked, which never happens. It can be proved by enumeration that, in 132 out of the 255 situations, spltClusters is guaranteed to be empty, which simplifies the computation of (5). Still, there are 123 situations that may cause cluster splitting. Figure 3 illustrates two such typical situations. Nevertheless, such situations do not certainly lead to cluster splitting, because the split parts observed in the 8-neighbor area may still be connected from outside. In this case, finding all the connected components of $Z \setminus \{c\}$ is

(a) (b)

Figure 3: Two typical situations of 8-neighboring cells that may cause cluster splitting if removing the cell in the center (in gray). Dark cells are marked. White cells are unmarked.

needed to obtain spltClusters. We restrict that if the cluster splitting does not result in at least one new cluster with discrepancy score greater than the original $\log L(Z)$, the cell will not be removed. The discrepancy gain function of removing operation, $G_r(Z, c)$, is thus defined as:

$$G_r(Z, c) = \max\{S_r(Z, c) - \log L(Z), 0\} \tag{6}$$

An inner-neighbor maximizing $G_r(Z, c)$ will be considered to remove from Z. The inner- and outer-neighbors are to be updated incrementally (lines 16–18) after the removal.

As an alternative to removing an inner-neighbor, growing an outer-neighbor is also evaluated. Key steps of such a cluster growing (lines 19–32) are similar to `LocateCluster` (Algorithm 3). Once a cell maximizing $G_g(Z, c)$ is determined to be included in Z, again, probable cluster merging is handled. The inner- and outer-neighbors are also to be updated (lines 23, 24, 27–32). Different from `LocateCluster` where inner-neighbors are not considered, a special case of updating inner-neighbors needs to be handled (line 29): Suppose within a cluster, there is a hole with a size of one cell, which must be an outer-neighbor[2]. If this hole is chosen for growing, then its 8-neighbor should be removed from inner-neighbors and no insertion is performed, which is different from the normal cases (line 31). By comparing the maximized $G_r(Z, c)$ and $G_g(Z, c)$, either removing or growing that better improves the discrepancy score is applied at a time. The refinement stops if there is no further improvement on the current level (line 8).

2.2 Computational Complexity Analysis

We consider the time complexity of detecting one cluster in `RefineScan` (Algorithm 2). Data aggregation (lines 1–2) costs $O(n + N^2)$. Building the grid hierarchy (line 3) costs $O((\frac{N}{2})^2 + \ldots + (\frac{N}{2^{l_{max}}})^2) = O(N^2 \cdot \frac{4^{l_{max}} - 1}{3 \cdot 4^{l_{max}}}) = O(N^2)$. The dominant computation on lines 4–8 is `LocateCluster` (Algorithm 3). The complexity of growing one cluster is of the same order of the cluster growing in GridScan given $(\frac{N}{2^{l_{max}}})^2$ grids [5], which can be proved to be $O(N^2)$.

In the refinement step (lines 9–15), the primary computations include line 11 that costs $O((\frac{N}{2^{(l-1)}})^2)$ and `RefineCluster` (Algorithm 5). Inside `RefineCluster`, the primary computations are lines 4–5. All the rest set operations can have efficient implementation so that their cost is dominated. We can infer that the size of inner- or outer-neighbors approximately equals the length of a cluster's perimeter P. On level l, in extreme cases, $P_l = O((\frac{N}{2^l})^2)$, whereas in most cases, $P_l = O(\frac{N}{2^l})$. Calling `GetInNB`(Z) in `RefineCluster` (line 1) at the first time after cluster growing on the top

[2]It can be proved that such a hole cannot be generated in upper levels since the unit cell size of upper levels must be larger. Therefore, it can only be generated on the current level. A theoretically possible cause can be that, a hole with area larger than a cell is generated by cluster merging, and it is filled by growing with one cell left unmarked.

level needs to traverse all outer-neighbors. But afterwards, if all inner- and outer-neighbors are always kept together with Z, `GetInNB`(Z) only costs $O(1)$. `GetOuNB`(Z) on line 2 always costs $O(1)$ because outer-neighbors are already maintained since `LocateCluster`. Finding c_{in} (line 4) costs $O(P_l \cdot (|\mathsf{spltClusters}| + p \cdot |Z|))$, where $O(|Z|)$ is the size of cluster Z (also is the complexity of finding connected components), and p is the probability of observing the 123 situations that may cause cluster splitting (see Section 2.1.2). We have $|\mathsf{spltClusters}| \leq 3$ and in most cases it is equal to 0. Besides, empirical results show that usually $p < \frac{123}{255}$. Therefore in normal cases, $O((\frac{N}{2^l})^3)$ can be a reasonable cost approximation for line 4, if assuming $|Z|$ is of order $O((\frac{N}{2^l})^2)$. Similarly, finding c_{out} (line 5) costs $O(P_l) = O(\frac{N}{2^l})$. Therefore, `RefineCluster` has a complexity of $O((\frac{N}{2^l})^3)$. The cost of refinement on all levels is thus $O(\sum_{l=0}^{l_{max}}((\frac{N}{2^{(l-1)}})^2 + (\frac{N}{2^l})^3)) = O(N^2 \cdot \frac{16 \cdot (4^{l_{max}}-1)}{3 \cdot 4^{l_{max}}} + N^3 \cdot \frac{8^{(l_{max}+1)}-1}{7 \cdot 8^{l_{max}}}) = O(N^3)$.

Overall, the time complexity of `RefineScan` is $O(n+N^2+N^3) = O(n+N^3)$. As can be seen, `RefineScan` can handle very large datasets without difficulty, because the complexity is linear to dataset size n with fixed N. Note that N may not be very large in practice (e.g., N=1K is very large already). If $n \gg N$, the complexity approaches $O(n)$.

Now let us compare a naive algorithm that maximizes the multi-scale discrepancy directly on the base level grids by exhaustive search. Given $N \times N$ grids, there are $2^{(N^2)}$ possible ways of marking them. The algorithm will cost $O(n+2^{(N^2)})$ to find the global optimum, which is unacceptable in practice. Although `RefineScan` employs a greedy search that does not guarantee the global optimality, as a trade-off, its computational cost is significantly lower. Experiments in the next section will show that `RefineScan` can have remarkable performance when compared with the state-of-the-art discrepancy maximization algorithms.

3. EXPERIMENTAL STUDIES

In this section, we will experimentally evaluate the performance of `RefineScan` using a variety of public datasets. We will also study the impacts of parameters on `RefineScan` as well as the algorithm's scalability.

3.1 Tasks, Datasets, and Common Settings

Eight overdensity and underdensity detection tasks are conducted on six datasets (see Table 2). More or less, these datasets all contain multi-scale spatial discrepancies, which will be clearly seen later. These datasets were also used as benchmarks in previous studies [3, 4, 5]. All the tasks are overdensity detections except for task 3. *Dataset1* and *Dataset2* are purely spatial datasets. "*Emerging*", "*expanding*", and "*moving*" are spatio-temporal datasets. We adopt certain data chunks of two weeks from the original datasets, with data collected in the earlier week treated as controls and data collected in the latter week treated as cases. *Epidemic* is also a spatio-temporal dataset, and each record is a microblog with time stamp and GPS location. The original dataset containing totally 1,023,077 microblogs is from the IEEE VAST Challenge 2011 Mini-Challenge 1[3]. *Fever* and *Diarrhea* are its subsets, corresponding to two typical subgroups of microblogs that are classified by epidemic keywords fever and diarrhea. In the data chunk of May19day, microblogs containing the certain keywords are treated as cases, and the rest are treated as controls. All the data preparations (including how the data are grouped and labeled) are exactly the same as in the literature. More details of these datasets, the ground truths, and the pre-processing steps can be found in [3, 5].

[3] `http://hcil.cs.umd.edu/localphp/hcil/vast11`

Table 2: Cluster detection tasks

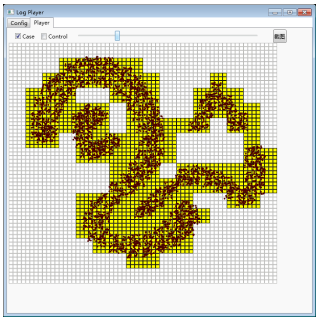
#task	dataset	task type
1	Dataset1	overdensity
2	Dataset2	overdensity
3	Dataset2	underdensity
4	"emerging": weeks 3–4	overdensity
5	"expanding": weeks 3–4	overdensity
6	"moving": weeks 4–5	overdensity
7	Epidemic - Fever: May19day	overdensity
8	Epidemic - Diarrhea: May19day	overdensity

Based on the common discrepancy function, $\log L(Z)$, we compare `RefineScan` with a typical grid-based algorithm, GridScan [5], Kulldorff's spatial scan statistic [9] using circular and elliptic windows (*SaTScan* software [11] as implementation), and Neill's scan statistic [12] using rectangular window on regular grids (*city4_applic* software [13] as implementation). In Neill's approach, only overdensity detection with Poisson model is implemented. On *Dataset2* and *Epidemic*, Poisson model is used for all algorithms. On the other datasets, Bernoulli model is used for all algorithms except Neill's approach. Default minP =1 is used in `RefineScan`. Unless explicitly mentioned, parameters of the candidate algorithms are set to their default values. We use $K = 99$ Monte Carlo simulations and report clusters significant at 0.05 level. All experiments are done on a 64-bit Win 7 machine with Intel Core 2 2.66GHz CPU plus 3G memory.

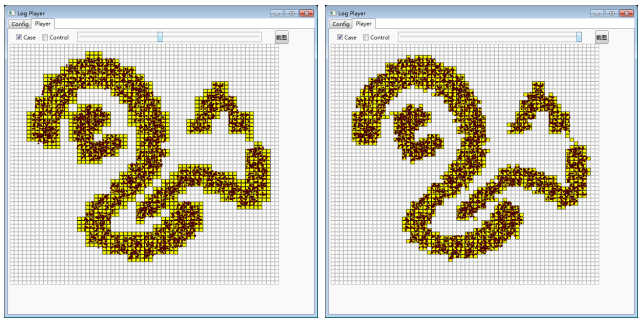
We will first evaluate the visualization results of detected clusters for each task. This verifies the necessity of characterizing multi-scale spatial discrepancy. Then, we will evaluate the numerical results by comparing the maximized discrepancy scores and analyzing the CPU time cost of `RefineScan`. At last, we will analyze the impact of parameters on `RefineScan`, and study the algorithm's scalability.

3.2 Visualization Evaluation

On task 1, we use N=64, l_{max}=2 in `RefineScan`. The grid hierarchy is thus a 3-level structure with 16×16 top level grids. As a fair comparison, we also test N=16 and 64, respectively, for both GridScan and Neill's approach. Data distribution of Dataset1 is shown in Figure 1(a). As an illustration of the execution process of `RefineScan`, Figure 4 shows the snapshots of the cluster's shape outputted by `RefineScan` (highlighted grids in the figure) on each level of the grid hierarchy. We can see that at first, `RefineScan` locates all the overdensity regions on the top level (level 2) with large connected grids, and then refines the overall shape with smaller grid units on the lower levels (levels 1 and 0) while keeping the connectedness on larger scales. The top-down procedure of cluster refinement can be clearly seen. The comparison of all candidate algorithms' results on task 1 are summarized in Figure 5. The results of `RefineScan` and GridScan are shown by highlighted grids, SaTScan results are shown by circles and ellipses, and the results of Neill's approach are shown by rectangles in dashed blue line. We can see that `RefineScan` better characterizes the cluster shape than the others. GridScan cannot well approximate the cluster boundary with large grids although the correct regions are identified. Also, it cannot detect significant overdensity at larger scales with too small grids: Some regions

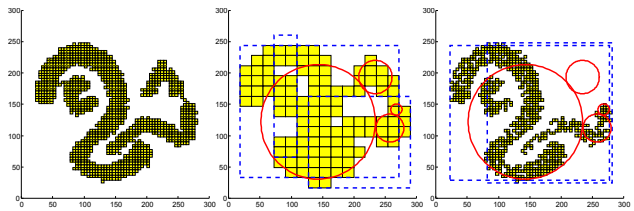


(a) LocateCluster on level 2



(b) Refinement on level 1 (c) Refinement on level 0

Figure 4: Illustration of RefineScan execution on task 1. Clusters are shown with base level grids. The *control* data are not displayed for clarity.



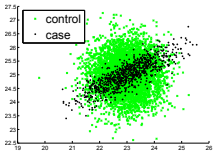
(a) RefineScan (b) Others N=16 (c) Others N=64

Figure 5: Results on Dataset1 (task 1)

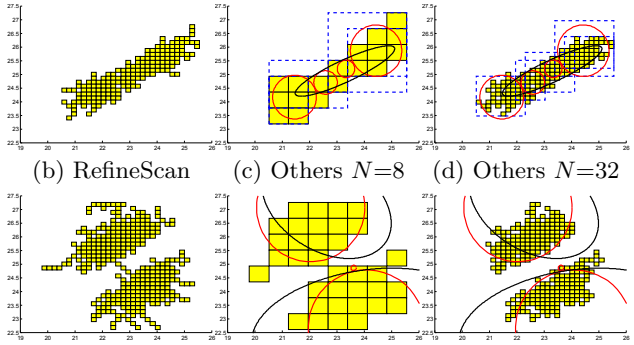
with dense cases easily regarded insignificant. It is noteworthy that, although the finest grids used in `RefineScan` are the same as in GridScan (64×64), their results are far different. The effectiveness of `RefineScan` in terms of handling multi-scale spatial discrepancy is clearly demonstrated. Neither SaTScan nor Neill's approach can work well: Due to the limitation of the regular scanning windows, they cannot recognize the true cluster shape and often wrongly detect a large connected region as multiple (and overlapping) ones.

On tasks 2 and 3, we use N=32, l_{max}=2 in `RefineScan`. Therefore, the grid hierarchy is 3-level with 8×8 grids on top. We also test N=8 and 32 for GridScan and Neill's approach. The dataset and cluster results are shown in Figure 6. We can see that the comparisons generate similar conclusions to task 1. `RefineScan` better characterizes the cluster's overall shape. In contrast, other algorithms either split a connected region into separate ones or report unnecessarily large areas. Still, with the same finest grids (32×32), `RefineScan` behaves differently from GridScan. The advantage of `RefineScan` will be further illustrated when examining the maximized discrepancy scores in Section 3.3 (Table 3).

On tasks 4–6, we use N=40, l_{max}=1 in `RefineScan`, meaning a grid hierarchy of 2 levels with 20×20 grids on top is employed. Results of the other candidates are adopted from [5]



(a) Dataset2



(b) RefineScan (c) Others N=8 (d) Others N=32

(e) RefineScan (f) Others N=8 (g) Others N=32

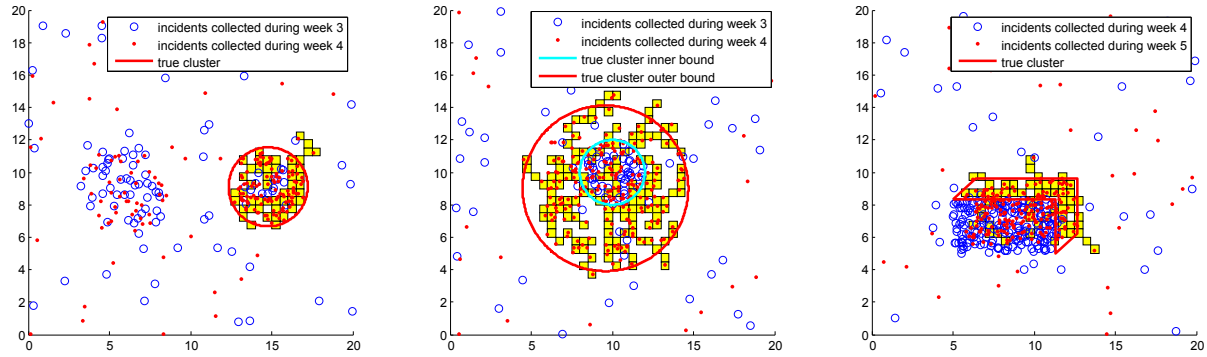
Figure 6: Dataset2 and results. Task 2 results are shown in (b)(c)(d) and task 3 results in (e)(f)(g).

for comparison. GridScan and Neill's approach use N=20. The results are shown in Figure 7. The data in these datasets are not as dense as in Dataset1 and Dataset2, which makes it difficult to capture the patterns with small grids. Moreover, 20×20 is an appropriate setting considering the scales in which the true clusters are generated [3]. The true clusters in the datasets are a circle, a large-scale ring, and an L-shaped area, respectively. Again, we can find that `RefineScan` captures the correct spatial discrepancy structures, improving the results of GridScan. More detailed numerical improvements can be found in Section 3.3 (Table 3). On the other hand, SaTScan and Neill's approach do not work well, even when facing the circular true cluster (see Figure 7(d)).

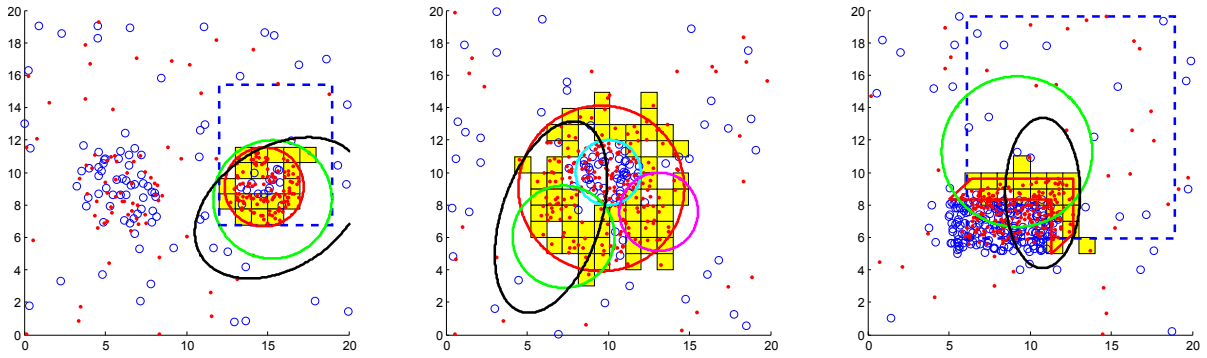
On tasks 7 and 8, we only compare `RefineScan` with GridScan since the true clusters are notably irregular. 30×76 base grids and l_{max}=1 are used in `RefineScan`. The upper level grids are thus 15×38. In GridScan, 15×38 and 30×76 grids are used, respectively. Results are visualized in Figure 8. We can see that `RefineScan` again works better: One cluster for fever and one cluster for diarrhea are detected, which is consistent with the ground truth (c.f. [5]), and the shapes are correctly characterized with plenty of details. GridScan can have satisfactory results with coarse grids. But if using fine grids, the diarrhea cluster breaks into four pieces, and the fever cluster breaks into two parts with much smaller sizes. This again indicates that as a single-scale algorithm, GridScan can be sensitive to the grid setting, whereas `RefineScan` is more robust due to combining and utilizing the information from multiple spatial scales.

3.3 Numerical Evaluation

The maximized discrepancy scores obtained by each candidate algorithm on the eight tasks are reported in Table 3. We can find that `RefineScan` performs the best throughout the experiments. It always finds the largest discrepancy score, indicating its remarkable performance in maximizing multi-scale spatial discrepancy from a variety of datasets even in presence of irregularly shaped clusters. Since the search strategies of the candidate algorithms differ with each other and are implemented in different languages and programming models, it is not meaningful to directly compare



(a) RefineScan on "emerging" (b) RefineScan on "expanding" (c) RefineScan on "moving"



(d) Others on "emerging" (e) Others on "expanding" (f) Others on "moving"

Figure 7: Results on the three spatio-temporal datasets (tasks 4–6). In all the figures, true clusters are explicitly denoted. RefineScan results are shown in (a)–(c). Results of other algorithms are shown in (d)–(f).

Table 3: Comparisons of maximized discrepancy scores. RefineScan performs the best (bolded). For GridScan and Neill's approach, when two grid sizes are used for a task, the results obtained by coarse grids (smaller N) are shown in the upper row.

#	RefineScan	GridScan	Neill's approach	SaTScan (circular)	SaTScan (elliptic)
1	**3338.26**	2202.52 1074.77	539.569 686.293	392.750	N/A$^\sharp$
2	**299.976**	219.939 292.687	109.868 91.575	61.042	142.031
3	**277.805**	178.414 137.935	N/A†	126.615	137.059
4	**50.1049**	36.3827	9.55162	30.7689	31.4163
5	**89.0435**	49.6349	0*	12.2105	17.0193
6	**63.0508**	33.9758	15.6311	26.8212	28.4956
7	**240.479**	160.343 113.954	–	–	–
8	**401.706**	331.344 181.249	–	–	–

$^\sharp$Insufficient memory. †Underdensity detection not supported.
*No cluster was detected.

their computational time. Therefore, here we only report the CPU time of `RefineScan` on each task (one run of Algorithm 2 on the original data without Monte Carlo simulations and single threaded C++ implementation) in Figure 9. We can find that generally, larger dataset size (n), the number of base grids (N), l_{max} and true cluster size (i.e., the number of marked grids), and a more irregularly shaped multi-scale discrepancy, will result in more CPU time. Overall, the computational cost of `RefineScan` is moderate.

3.4 Impact of Parameters and Scalability

To analyze the impacts of the primary parameters N and l_{max} on `RefineScan`, and further verify the algorithm's scalability, we conduct additional experiments on a simulated dataset, D1K [5] that contains 10^3 cases and 10^3 controls for Bernoulli model based overdensity detection. It is noteworthy that although the reported result here is on a specific dataset, the conclusion coincides with the theoretical analysis in Section 2.2, and can be generalized to any datasets. We first test the performance of `RefineScan` using feasible combinations of $N \in \{8, 16, 32, 64, 128\}$ and $l_{max} \in \{0, 1, 2, 3, 4, 5, 6\}$. Note that l_{max} is bounded by $\log_2 N$, and in practice at most $\log_2 N - 1$ since a search starting with only one grid unit on the top level is meaningless in most cases. Therefore, only the combinations satisfying $l_{max} < \log_2 N$ are tested.

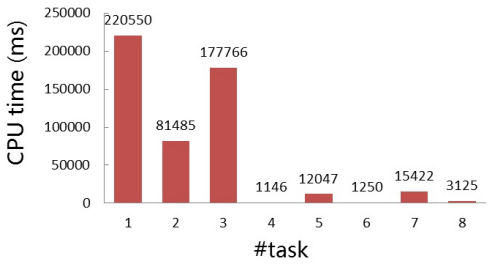
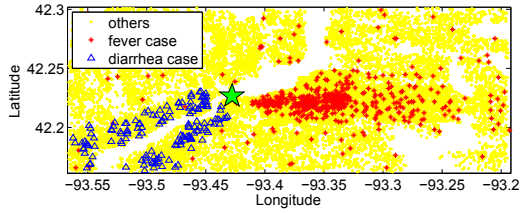
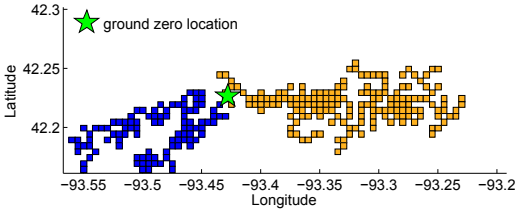


Figure 9: CPU time of RefineScan on each task

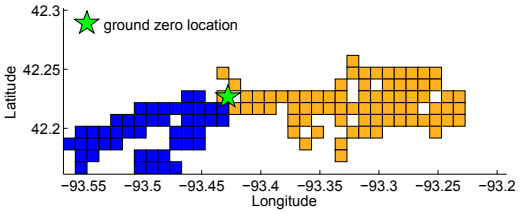
The results of maximized discrepancy scores are summarized in Table 4. In the table, the largest scores obtained are shown in bold. We can see that in general, when l_{max} is sufficiently large, a larger N leads to a larger discrepancy score. This is natural since a finer base granularity helps better characterize clusters' shape. Meanwhile, with a fixed N, l_{max} is not necessarily to be as large as $\log_2 N - 1$ to obtain the maximized discrepancy score. This is also reasonable, because as long as the grid size on a level of the



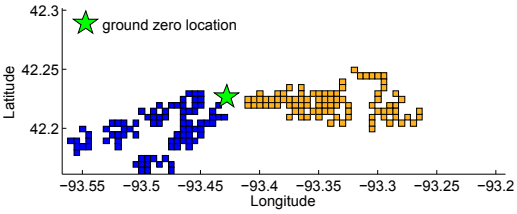
(a) Data Overview



(b) RefineScan



(c) GridScan: 15×38



(d) GridScan: 30×76

Figure 8: Results on tasks 7 and 8. Diarrhea clusters are shown in blue. Fever clusters are shown in orange.

Table 4: Maximized discrepancy scores with varying l_{max} and N

l_{max} \ N	8	16	32	64	128
0	57.2237	790.295	278.854	269.319	102.171
1	516.732	790.295	1016.2	336.941	327.793
2	57.2237	790.295	1016.2	1164.41	397.031
3	-	790.295	1016.2	1164.41	**1265.07**
4	-	-	1016.2	1164.41	**1265.07**
5	-	-	-	1164.41	**1265.07**
6	-	-	-	-	**1265.07**

Table 5: CPU time cost (in second) with varying l_{max} and N

l_{max} \ N	8	16	32	64	128
0	0.031	0.125	0.266	0.797	2.263
1	0.089	0.36	7.942	13.672	53.879
2	**0.026**	0.968	8.219	65.754	129.106
3	-	**0.343**	8.985	67.969	557.234
4	-	-	**8.078**	69.453	562.656
5	-	-	-	**67.406**	565.094
6	-	-	-	-	**600.641**

grid hierarchy is sufficiently large to capture the global scale structure, an even larger grid size at a higher level does no more help. In contrast, if N is too small (e.g., N=8, the first column of Table 4), meaning the base level grids are still too coarse, the algorithm may become unstable due to missing details of the data distribution in the aggregation step. Besides, even with a sufficiently large N, if l_{max} is too small (e.g., $N = 128$ and $l_{max} \in \{0, 1, 2\}$ in Table 4), `RefineScan` cannot benefit much from the grid hierarchy due to missing data distribution information at larger scales. Hence, `RefineScan` cannot work well either.

The CPU time cost of `RefineScan` (still, one run without Monte Carlo simulations) is summarized in Table 5. Generally, when the dataset size n is constant, larger N and larger l_{max} values naturally result in increased CPU time. Section 2.2 showed that `RefineScan` has a computational complexity of $O(n + N^3)$, where the $O(n)$ complexity is introduced only by parsing the dataset for one time aggregation before detecting clusters. Therefore, as validation on the linearity with n is trivial, we omit it here. We only need to verify that with a constant n, the primary computation is linear to N^3. Accordingly, we choose the CPU time cost obtained by the largest l_{max} for each N (values are bolded in Table 5) and study the relationship between the real CPU time and N. Figure 10 shows the samples and a fitted

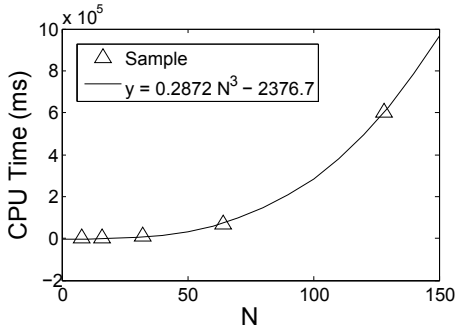


Figure 10: CPU time against N

curve from the samples. The curve is obtained by regressing the CPU time with variable N^3 using a linear model $y = aN^3 + b$. When plotting the linear model against N, it becomes a cubic curve. The obtained linear equation is $y = 0.2872N^3 - 2376.7$ with R square 1.0, indicating a perfect linearity between the CPU time and N^3.

The results show the performance of `RefineScan` with changing parameters and verify the theoretical analyses in earlier sections. The guidelines of setting the parameters of `RefineScan` when applying it to new real-world problems can also be given: N should be set as large as possible if more detailed discrepancy structure is of great interest. However, this may lead to a larger computational cost (linear to N^3 given a dataset), and thus a trade-off should be considered in practice. The grid hierarchy plays a key role in successfully characterizing multi-scale discrepancies, while a large enough l_{max} (but may not necessarily close to $\log_2 N$ depending on the data distribution) is also helpful.

4. RELATED WORK

Previous studies on spatial discrepancy maximization for overdensity/underdensity detection either only consider regularly shaped clusters or just focus on a fixed spatial scale. Typical algorithms addressing regular shapes include the spatial scan statistics methods using circular and elliptic scanning windows proposed by Kulldorff et al. [9, 10] and the rectangular window based scan statistics proposed by Neill et al. [12, 14] and by Agarwal et al. [1]. Given a dataset with size n, the computational complexity of cluster detection (with excluding data preparation) of Kulldorff's scan

statistic methods is at least $O(n^2)$ when running without aggregation and becomes $O(N^2)$ when data are aggregated to $N \times N$ grids [9, 10]. It can be inferred that a naive search for rectangular regions costs $O(N^4)$ given $N \times N$ grids. Neill et al. reduces the complexity to $O((N \log N)^2)$ [12]. Agarwal et al's ϵ-approximation algorithm costs $O(\frac{1}{\epsilon} N^3 \log N)$ [1]. Compared with these algorithms, the proposed `RefineScan` in this paper is much more flexible and more powerful, and its computational complexity $O(N^3)$ is quite acceptable. There are also algorithms that can detect irregularly shaped clusters, but most of which work on a pre-defined spatial scale. Typical algorithms in this category include GridScan proposed by Dong et al. [5], the random-work based FS^3 proposed by Janeja and Atluri [8], AMOEBA proposed by Aldstadt and Getis [2], a flexible scan statistic proposed by Tango and Takahashi [17], etc. As analyzed in previous sections, because there is no explicit consideration on the multi-scale spatial structure embedded in data, these algorithms cannot well handle multi-scale spatial discrepancy. Moreover, their performance can be sensitive to the aggregation. Many of these algorithms employ exhaustive search, making their scalability poor. The lowest computational complexity by far is $O(N^2)$ of GridScan [5], which is linear to the number of grids ($N \times N$) or say aggregation areas. By comparison, we can see that `RefineScan` does not introduce heavy additional computations considering the significant performance improvement over these single-scale algorithms.

The traditional clustering algorithms in data mining can also be applied to spatial datasets to discover clustering patterns. For instance, DBSCAN proposed by Ester et al. [6] is capable of considering multi-scale information within the classical clustering framework. There are also grid-based algorithms, such as STING proposed by Wang et al. [18] and WaveCluster proposed by Sheikholeslami et al. [16]. A grid hierarchy is also employed in STING to store the multi-scale statistical information for spatial query speed-up. Nonetheless, as already discussed in Section 1, due to the intrinsic difference between the classical clustering problem and the overdensity/underdensity detection problem on which this paper focuses, these algorithms do not work on discrepancy maximization. Not to mention that `RefineScan` utilizes the grid hierarchy in a different way: The top-down manner of the cluster refinement in `RefineScan` essentially implements the information diffusion across multiple spatial scales, which is totally different from the speed-up purpose motivating the grid hierarchy in STING.

5. CONCLUSION

In this paper, we propose a novel algorithm `RefineScan` for multi-scale spatial statistical discrepancy maximization. By employing a multi-level grid hierarchy with efficient cluster growing and refinement methods, `RefineScan` is good at maximizing flexible spatial discrepancies even when facing multi-scale and irregularly shaped discrepancy structures. Experiments show that `RefineScan` outperforms the state-of-the-art algorithms by (1) finding the largest discrepancy scores throughout the experiments, and (2) better characterizing the true shapes of overdensity and underdensity. The impacts of parameters on `RefineScan` are also studied. Guidelines of setting the parameters are given accordingly. Given $N \times N$ base level grids and a dataset of size n, the computational complexity of `RefineScan` is $O(n + N^3)$, where N can be bounded to a small number in practice. It makes `RefineScan` a scalable and effective algorithm for detecting spatial anomalous events from large datasets.

6. REFERENCES

[1] D. Agarwal, A. McGregor, J. Phillips, S. Venkatasubramanian, and Z. Zhu. Spatial scan statistics: approximations and performance study. In *KDD'06*, pages 24–33. ACM, 2006.

[2] J. Aldstadt and A. Getis. Using AMOEBA to create a spatial weights matrix and identify spatial clusters. *Geographical Analysis*, 38(4):327–343, 2006.

[3] W. Chang, D. Zeng, and H. Chen. A stack-based prospective spatio-temporal data analysis approach. *Decision Support Systems*, 45(4):697–713, 2008.

[4] W. Dong, X. Zhang, Z. Jiang, W. Sun, L. Xie, and A. Hampapur. Detect irregularly shaped spatio-temporal clusters for decision support. In *SOLI*, pages 231–236. IEEE, 2011.

[5] W. Dong, X. Zhang, L. Li, C. Sun, L. Shi, and W. Sun. Detecting irregularly shaped significant spatial and spatio-temporal clusters. In *SDM*, pages 732–743, 2012.

[6] M. Ester, H. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, pages 226–231, 1996.

[7] V. Iyengar. On detecting space-time clusters. In *KDD'04*, pages 587–592. ACM, 2004.

[8] V. Janeja and V. Atluri. Random walks to identify anomalous free-form spatial scan windows. *IEEE Trans. Knowl. Data Eng.*, 20(10):1378–1392, 2008.

[9] M. Kulldorff. A spatial scan statistic. *Comm. Statist. Theory Methods,*, 26(6):1481–1496, 1997.

[10] M. Kulldorff, L. Huang, L. Pickle, and L. Duczmal. An elliptic spatial scan statistic. *Statistics in Medicine*, 25(22):3929–3943, 2006.

[11] M. Kulldorff and Information Management Services, Inc. SaTScan^{TM} v8.0: Software for the spatial and space-time scan statistics, 2009. `http://www.satscan.org`.

[12] D. Neill and A. Moore. Rapid detection of significant spatial clusters. In *KDD'04*, pages 256–265, 2004.

[13] D. Neill, A. Moore, K. Daniel, and R. Sabhnani. city4_applic software for Scan Statistics, 2011. Auton Lab, Carnegie Mellon University, `http://www.autonlab.org/autonweb/downloads/software.html`.

[14] D. Neill, A. Moore, M. Sabhnani, and K. Daniel. Detection of emerging space-time clusters. In *KDD'05*, pages 218–227, 2005.

[15] S. Openshaw. *The modifiable areal unit problem.* Geo Books, 1984.

[16] G. Sheikholeslami, S. Chatterjee, and A. Zhang. Wavecluster: A multi-resolution clustering approach for very large spatial databases. In *VLDB*, pages 428–439, 1998.

[17] T. Tango and K. Takahashi. A flexibly shaped spatial scan statistic for detecting clusters. *International Journal of Health Geographics*, 4(1):11, 2005.

[18] W. Wang, J. Yang, and R. Muntz. STING: A statistical information grid approach to spatial data mining. In *VLDB*, pages 186–195, 1997.