Detecting Irregularly Shaped Significant Spatial and Spatio-Temporal Clusters

Weishan Dong Xin Zhang Li Li Changhua Sun Lei Shi Wei Sun *

Abstract

Detecting significant overdensity or underdensity clusters in spatio-temporal data is critical for many real-world applications. Most existing approaches are designed to deal with regularly shaped clusters such as circular, elliptic and rectangular ones, but cannot work well on irregularly shaped clusters. In this paper, we propose GridScan, a grid-based approach for detecting irregularly shaped spatial clusters. In GridScan, a cluster is asymptotically described by a set of connected grid cells and is computed by a fast greedy region-growing algorithm with elaborating cluster merging in the process. The time complexity of GridScan is linear to the number of grids, making it scalable to very large datasets. A prospective spatio-temporal cluster detection approach, GridScan-Pro, is also proposed by extending GridScan. Experiments and a case study in the epidemic scenario demonstrate that our approaches greatly outperform existing ones in terms of accuracy, efficiency, and scalability.

1 Introduction

Given baseline information, detecting significant *overdensity* or *underdensity* clusters is an important problem in geographical data mining and spatio-temporal data analysis. There are numerous applications of such kind of cluster detection. For example, given the population distribution of a city, detecting the outbreak area of a possible epidemic where the number of disease cases is significantly larger than expected, or identifying crime hotspots/coldspots where crime rates are unusually high/low. In these scenarios, the data analyzed, such as disease cases and crime incidents, composed of a set of geo-referenced points in purely spatial analysis, and can also be a point process in spatio-temporal analysis. The concept of cluster here is defined as a local geographic region associated with a density measure, whose value is significantly higher or lower than expected given the baseline. Such a cluster indicates the overdensity or underdensity anomaly. The unusualness of a density value is measured by its p-value. If the p-value is smaller than a significance level (e.g., 0.05 or 0.01), it implies that the density can rarely be observed and the cluster is statistically significant. The cluster detection problem involves finding the location and extent of a cluster by maximizing the density measure and calculating its p-value.

In this paper, we consider the detection of such clusters from a point dataset. Specifically, we focus on the irregularly shaped clusters constituting the most parts in real-world applications. For example, the cone-shaped cluster indicating a disease spread pattern observed in the epidemic case. Also, more accurate detection of clusters beyond the regularly shaped ones can help to achieve more efficient emergency response and better situational awareness, which could be of crucial importance for rapid response to threats of bioterrorism and major infectious disease outbreaks (e.g., West Nile Virus).

Despite of the importance of detecting the irregularly shaped clusters, existing cluster detection approaches [10, 11, 16, 18, 3] for the point data are generally designed to only deal with the regularly shaped clusters (e.g., circle, ellipse, or rectangle), primarily due to computational reasons. When applied to detecting irregularly shaped clusters, their performance is quite limited. Meanwhile, the classical clustering algorithms in the data mining field dealing with irregularly shaped clusters, such as DBScan [7, 2], STING [24], and WaveCluster [22], also cannot work in the scenarios considered here due to the specialty of this problem. The cluster detection problem on which this paper focuses is different from the classical clustering problem. Later in Section 3, the objective function of overdensity/underdensity detection will be given, including how baseline information is involved. The difference with the goal of classical clustering, which is to maximize both the homogeneity within each cluster and the heterogeneity among different clusters, can be seen clearly. To be specific, first, classical clustering algorithms generally discover clustering pattern, but not the overdensity defined on an explicit baseline. Second, these algorithms cannot detect the underdensity cluster in general. Third, because the baseline is not explicitly considered, the p-values of detected clusters, which directly help decision makers to understand the unusualness of results, are ignored in most cases of classical clustering.

To remedy the shortcomings of previous ap-

*IBM Research – China, 19 Zhongguancun Software Park, 8 Dongbeiwang West Road, Beijing 100193, PR China. Email: {dongweis,zxin,lilichina,schangh,shllsh,weisun}@cn.ibm.com

proaches, in this paper, we propose a grid-based cluster detection approach, namely GridScan, which is capable of detecting irregularly shaped spatial clusters. The algorithmic flow of GridScan can be summarized into three steps. 1) Search for local regions with maximum density. The 2D geographical space is divided into, say, $N \times N$ regular grids. Data points and baseline information is spatially aggregated to the grid cells. A greedy region-growing algorithm is employed to find the connected regions of cells, which can asymptotically describe any irregular shape to accurately characterize the true clusters at the given spatial granularity. The density values of clusters are calculated in the iterations of the region growing. 2) Perform statistical hypothesis test to calculate the p-value of the density. Monte Carlo simulation is used to test against the null hypothesis of no clustering. The smaller the p-value is, the more likely the corresponding cluster is significant. 3) Report the clusters whose p-value is smaller than a given significance level (e.g., 0.05 or 0.01).

The key challenge in designing GridScan is to reduce the time complexity in the process of growing irregularly shaped clusters in the step 1). Existing grid-based cluster detection approaches [16, 1] are only capable of finding rectangular clusters, however their time complexity is still high. Given $N \times N$ grids, naive search for rectangular regions costs $O(N^4)$. Neill et. al [16] reduces the search time to $O((N \log N)^2)$. Agarwal et. al's ϵ-approximation algorithm [1] is of order $O(\frac{1}{\epsilon}N^3 \log N)$. Comparatively speaking, GridScan is far more efficient that it only costs $O(N^2)$, and it is capable of finding irregularly shaped cluster. Another challenge lies in the characteristic of the density measure and the greedy nature of the local search, there is theoretical possibility that a cluster grown in later iterations needs to be merged with a previously grown cluster to obtain a larger density. This makes the cluster growing more complex than a standard region-growing algorithm. With properly dealing with this issue, GridScan is not only efficient but also effective.

So far in literatures, the only cluster detection approach which can detect clusters of irregular shape for point data is the SVM based approach proposed by Chang et. al [3], namely PSVC. However, it has a drawback that setting its parameters is hard. It is well known in the SVM community and recent studies on SVM based spatio-temporal analysis (e.g., [5]) have also shown that the kernel width and the margin parameters significantly influence the results. The facts that 1) these parameters do not have explicit physical implications, and 2) cluster detection is unsupervised, bring difficulties in finding appropriate settings of these parameters. Contrarily, as we will see, parameters of GridScan have clear physical implications and thus are easier to set. Empirical comparison can show that GridScan has similar capability of characterizing irregularly shaped clusters to PSVC. But as a non-grid-based algorithm, computational complexity of the SVM approach is relatively high, which is $O(n^2)$ in finding clusters (n is the dataset size) [3]. When facing extremely large datasets where usually $N \ll n$, GridScan has its advantage in terms of efficiency and scalability.

We also extend GridScan to a prospective spatio-temporal cluster detection approach, GridScan-Pro, by a simple yet effective prospective surveillance framework. It can accurately detect irregularly shaped significant data change at early stages.

The rest of this paper is organized as follows. Related work is reviewed in Section 2. We propose GridScan in Section 3. Experimental studies are given in Section 4. In Section 5, a case study on epidemic outbreak detection is conducted. Finally, conclusions are drawn in Section 6.

2 Related Work

For spatial cluster detection, the search for a region is only on a 2D geographical space. In existing approaches, usually a regularly shaped scanning window is used to scan over the 2D space. Kulldorff [10] proposed spatial scan statistic using circular window to exhaustively test every circular region. Neill et. al [16, 18] and Agarwal et. al [1] proposed methods of searching for axis-aligned rectangular regions. Conley et. al [4] used genetic algorithm (GA) to find clusters shaped as ellipses with axes oriented to vertical and horizontal axes. Kulldorff et. al [13] applied elliptic window for the spatial scan statistic, but with finite rotations and length ratios of semimajor axis to semiminor axis. Sahajpal et. al [21] proposed a GA to find clusters shaped as intersections of circles of different sizes and centers. Because of the shape limitation, we can imagine that it will be hard to accurately detect an L-shaped or ring-shaped cluster using these approaches.

When extending to spatio-temporal domain, the time dimension is added into the search space. A spatio-temporal cluster is described not only by a region, but also by a time interval. A 3D cylindric scanning window can be used as in the space-time scan statistic proposed by Kulldorff et. al [12, 11]. Iyengar [8] also proposed a heuristic search to find spatio-temporal clusters shaped as square pyramids. Another way is to segment the data into chunks by arrival time, and then analyze and compare the chunks to detect the anomaly [18, 3]. Spatio-temporal cluster detection can serve as a retrospective analysis tool to analyze historical data,

where a cluster has a start time and an end time. It can also be used for prospective surveillance in detecting events such as epidemic outbreaks at an early stage, where usually only the start time of a cluster is paid attention to.

Another branch of cluster detection approaches is designed only for areal data. Areal data means that the data are spatially aggregated to administrative areas whose shapes and boundaries are fixed, and the precise location information for each specific data case is unknown. Several irregularly shaped cluster detection approaches for areal data have been proposed, e.g., [20, 6, 23, 9]. Nevertheless, few approach for point data has been proposed. Because of the absence of areal boundary, it becomes difficult to decide a cluster's extent for point data. As mentioned, so far, the only approach capable of detecting irregularly shaped clusters for point data is based on SVM [3], which has been discussed in Section 1. As can be easily seen, GridScan can also apply to areal data when given the neighboring information of the predefined areas. Compared with existing approaches for areal data which often employ exhaustive search, although all the main operations are dealing with connected regions (areas), GridScan works in a more smarter way and therefore is much faster. Details will be given later.

The computational complexity of existing approaches is often high. Let n denote dataset size. Taken purely spatial cluster detection as an example and not considering the hypothesis testing step, exhaustive search in Kulldorff's spatial scan statistic using circular window costs $O(n^2)$ [14], and so does the SVM approach [3]. Using elliptic window can further enlarge the complexity. Empirically, GA and heuristic search based approaches are slow, and their mathematical time bound is usually hard to give. Time complexities of existing grid-based approaches have been discussed in Section 1. Although some progress on the computational complexity has been made [16, 1], the search is still restricted to rectangular clusters. Computational complexities of the approaches for areal data have rarely been studied rigorously. Usually exhaustive search is employed, and only empirical computation time of some approaches has been reported (e.g., [6, 23]).

3 GridScan

3.1 Density Measure & Hypothesis Testing

Lots of density measures can be used for cluster detection. In GridScan, we focus on one of the most commonly used density measures in prior arts, the likelihood function of the spatial scan statistic proposed by Kulldorff [10]. For each candidate region Z, a density measure, the likelihood $L(Z)$, is calculated. Its definition differs when different probabilistic model is used. Two most typical models are Bernoulli model and Poisson model. The Bernoulli model is usually applied when the events are binary, e.g., when study people with or without a specific disease. Therefore, under Bernoulli model, data are classified into two groups. The baseline data is called *control*, and the data to be analyzed is called *case*. The summation of cases and controls is the population. The Poisson model is used when the number of cases is Poisson distributed and the cases denote presence of some rare event. Obviously, under Poisson model, the expected number of cases in each region is proportional to its population.

Let n_Z denote the number of cases inside a region Z, n_G the global number of cases of the entire dataset, N_Z the number of population inside Z, and N_G the global number of population of the entire dataset. $L(Z)$ is defined as follows. When using the Bernoulli model,

$$L(Z) = \left(\frac{n_Z}{N_Z}\right)^{n_Z} \cdot \left(1-\frac{n_Z}{N_Z}\right)^{N_Z-n_z} \cdot \left(\frac{n_G-n_Z}{N_G-N_Z}\right)^{n_G-n_Z} \cdot \left(1-\frac{n_G-n_Z}{N_G-N_Z}\right)^{(N_G-N_Z)-(n_G-n_Z)} \quad (3.1)$$

When using the Poisson model,

$$L(Z) = \left(\frac{n_Z}{E(n_Z)}\right)^{n_Z} \cdot \left(\frac{n_G-n_Z}{n_G-E(n_Z)}\right)^{n_G-n_Z}, \quad (3.2)$$

where $E(n_Z) = \frac{n_G}{N_G} N_Z$ denotes the expected number of cases inside Z under the null hypothesis[1]. No matter which model is used, when search for overdensity, only Z with $\frac{n_Z}{N_Z} > \frac{n_G-n_Z}{N_G-N_Z}$ is examined, which means more cases are observed than expected under the null hypothesis. Oppositely, when search for underdensity, only Z with $\frac{n_Z}{N_Z} < \frac{n_G-n_Z}{N_G-N_Z}$ is examined. The most likely overdensity/underdensity cluster is then detected by directly finding $arg\max L(Z)$. Very often, $\log L(Z)$ is used instead of $L(Z)$. Both forms are widely adopted in prior arts [10, 12, 11, 13, 16, 18, 1, 8, 3, 6, 23]. We can see from (3.1) and (3.2) that actually there is no restriction on the shape of Z. We will propose our approach based on density measure $\log L(Z)$.

After obtained a cluster Z on the original dataset, in the hypothesis testing stage, its p-value is to be calculated. Because the exact distribution of $L(Z)$ is unknown, Monte Carlo simulation has to be used to calculate the p-value [10]. A number of, say, K, replicated datasets having the same underlying population and the

[1] Additional baseline information can be also involved. If variables such as age, sex, income, etc., are considered in a disease surveillance scenario, they can be treated as *covariates* and $E(n_Z)$ becomes the covariate adjusted expected number of cases inside Z under the null hypothesis [10, 14].

same number of cases are generated randomly under the null hypothesis, where the cases are assigned among the population uniformly by randomization. For each replicated dataset, the search for the most likely cluster is carried out. If the value of $L(Z)$ is among the highest k when compared with the maximum likelihoods of the K replicated datasets, then the p-value of cluster Z is $\frac{k}{K+1}$. K=999 is often used in literatures for 0.001 precision of p-value.

3.2 Algorithm Hereafter, we will adopt the notations in Section 3.1 and the followings: D, an input 2D point dataset, $|D| = N_G$ and among which there are n_G cases; N, the dimension of grids, data in D will be spatially aggregated to $N \times N$ regular grid cells; minPOI, the minimum number of point-of-interest (POI) in a cell (default value 1), which will be introduced later; K, the number of Monte Carlo simulation (default value 999); θ, the significance level of clusters (default value 0.05); $\mathcal{C}$, the set of all grid cells, where $|\mathcal{C}| = N^2$. Each grid cell $c \in \mathcal{C}$ has two properties, n_c and N_c, which denote the number of cases and the population, respectively, in c. $\forall c \in \mathcal{C}, N_c \geq n_c \geq 0$, and $n_G = \sum_{c\in\mathcal{C}} n_c$, $N_G = \sum_{c\in\mathcal{C}} N_c$. A cell has one of two possible states, marked or unmarked. Let $\mathcal{C}^-$ denote the set of all unmarked cells, $\mathcal{C}^- \subseteq \mathcal{C}$.

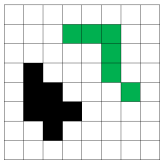


Figure 1: An example of 8×8 grids.

The aim of GridScan is to find a set of geographic regions, each of which denotes a cluster Z measured by $\log L(Z)$ and significant at level θ. A cluster Z is a set of marked and connected grid cells, $Z \subseteq \mathcal{C}$, $n_Z = \sum_{c\in Z} n_c$, $N_Z = \sum_{c\in Z} N_c$. Since $\mathcal{C}$ can be seen as an image with $N \times N$ pixels, we adopt the 8-connectedness in image processing field to define the connectedness between cells. Therefore, a cluster Z can also be seen as a connected region of marked cells. Whether a cell is marked and included in a cluster or not is determined by GridScan. An example of 8×8 grids is shown in Fig. 1. Black and green cells are marked, which constitute two connected regions according to the 8-connectedness. Other cells are unmarked.

The main steps of GridScan are shown in Algorithm 1. Lines 1–3 are the detection of all candidate clusters. Lines 5–9 are the Monte Carlo simulation. Then p-values of the candidate clusters are computed. Finally, only significant clusters are outputted. Function `FindClusters` on lines 3 and 8 is shown in Algorithm 2, which is to repeatedly call function `FindOneCluster` to find all potential clusters. `FindOneCluster` shown in Algorithm 3 is the core of GridScan. Briefly, it works like a region-growing algorithm that it first finds a seed cell c_0 to be the starting cell of a cluster Z, and then locally grows it by its 8-connected neighboring cells. The seed is chosen from the candidate seed set, candSeedSet, with maximum $\log L(\{c\})$, where set candSeedSet $\subseteq \mathcal{C}^-$ stores all candidate seed cells. When growing Z, one cell c_{next} is to be marked and included in Z each time, which is chosen from unmarked neighboring cells of Z with a maximum log likelihood gain $G(Z, c)$.

Algorithm 1: GridScan

```
   Input: D; N; minPOI; K; θ
   Output: A set of significant clusters, finalClusters
1  define C by N × N grid cells covering all points in D;
2  foreach c ∈ C do set n_c and N_c based on D;
3  candidateClusters ←FindClusters (C, minPOI);
4  simClusters ← ∅; finalClusters ← ∅;
5  for i ← 1 to K do
6      generate a replicated dataset D'_i;
7      foreach c ∈ C do set n_c and N_c based on D'_i;
8      tempClusters ← FindClusters(C, minPOI);
9      simClusters ←
       simClusters ∪ {arg max_{Z'∈tempClusters} log L(Z')};
10 foreach Z ∈candidateClusters do
11     k ← rank of log L(Z) comparing with
       log L(Z'), ∀Z' ∈simClusters;
12     p_Z ← k/(K+1);                    // p-value of cluster Z
13     if p_Z ≤ θ then finalClusters ←finalClusters ∪{Z};
14 return finalClusters;
```

Algorithm 2: FindClusters

```
   Input: C; minPOI
   Output: A set of clusters, clusterSet, each element of which
           is a cluster Z.
1  clusterSet ← ∅; C⁻ ← C; candSeedSet ← C⁻;
2  while candSeedSet ≠ ∅ do
3      {Z, candSeedSet, C⁻, clusterSet} ←
       FindOneCluster(candSeedSet, C⁻, clusterSet, minPOI);
4      if Z ≠ ∅ then clusterSet ← clusterSet ∪ {Z} else break;
5  return clusterSet;
```

Before defining $G(Z, c)$, we first define function $S(Z, c)$ indicating the supposed log likelihood if grow Z by cell c:

$$S(Z, c) = \log L(Z \cup \{c\} \cup \text{connClusters}) \ , \tag{3.3}$$

where connClusters $= \{Z' | c \in$ `GetNeighbors`$(Z'), Z' \in$ clusterSet$\}$ denotes which previously detected clusters will be connected with Z into one merged cluster if c is marked. Function `GetNeighbors`(Z) returns all 8-connected neighboring cells of Z, and clusterSet stores all previously detected clusters. We can easily find that |connClusters| ≤ 3 under the 8-connectedness. $G(Z, c)$ is further defined as:

$$G(Z, c) = \begin{cases} 0 & \text{if } \exists Z' \in \text{connClusters, s.t.} \\ & \quad S(Z, c) < \log L(Z'), \\ S(Z, c) - \log L(Z) & \text{otherwise.} \end{cases} \tag{3.4}$$

Since the cluster growing in GridScan is obviously local and greedy, it is possible that the boundary

Algorithm 3: FindOneCluster

Input: A set of candidate seed cells, candSeedSet; $\mathcal{C}^-$; Detected clusters in previous iterations, clusterSet; minPOI
Output: A cluster Z, which is a set of connected and marked cells; Updated candSeedSet; Updated $\mathcal{C}^-$; Updated clusterSet

```
1  Z ← ∅;
2  c_0 ← arg max_{c∈candSeedSet} log L({c});          // find seed cell
3  if log L({c_0}) > 0 and AboveMinPOI (c_0, minPOI) then
4      C⁻ ← C⁻ \ {c_0};                               // mark seed cell
5      Z ← {c_0};                                     // initialize Z
6      neighbors ← GetNeighbors(Z);
7      while true do
8          c_next ← arg max_{c∈neighbors} G(Z, c);
9          if G(Z, c_next) > 0 and AboveMinPOI(c_next, minPOI)
           then
10             connClusters ← {Z'|c_next ∈
               GetNeighbors(Z'), Z' ∈ clusterSet};
11             foreach Z' ∈ connClusters do
12                 Z ← Z ∪ Z';                        // merge with Z
13                 neighbors ← neighbors ∪ GetNeighbors(Z');
14                 clusterSet ← clusterSet \ {Z'};
15             C⁻ ← C⁻ \ {c_next};                    // mark cell c_next
16             Z ← Z ∪ {c_next};                      // grow Z by c_next
17             neighbors ← neighbors \ {c_next} ∪
               (GetNeighbors({c_next}) ∩ C⁻);
18         else
19             candSeedSet ← candSeedSet \ (Z ∪ neighbors);
               // since Z can no longer grow from neighbors,
                  next round seed cell selection won't be
                  from neighbors, either
20             break;                                 // stop growing
21 return {Z, candSeedSet, C⁻, clusterSet};
```

Algorithm 4: AboveMinPOI

Input: A grid cell, c; minPOI
Output: A boolean value, *true* or *false*

```
1 switch cluster detection objective do
2     case overdensity return (n_c ≥ minPOI);
3     case underdensity return (N_c − n_c ≥ minPOI);
```

of a newly growing cluster Z can approach to one or more (at most three) previously detected clusters in clusterSet, which have stopped growing locally in previous iterations. Suppose at some time, a cell $c \in$ GetNeighbors$(Z) \cap$ GetNeighbors(Z') is being considered to be marked and included in Z, where $Z' \in$ clusterSet. Apparently, marking c will connect clusters Z and Z' into one and produce a merged cluster $Z'' = Z \cup \{c\} \cup Z'$. However, $\log L(Z'') = S(Z, c)$ is not surely greater than both $\log L(Z)$ and $\log L(Z')$. In GridScan, if $\log L(Z'') = S(Z, c) < \log L(Z')$, c will not be marked. In other words, GridScan restricts the log likelihood of every potential cluster to increase monotonously. Merging Z with existing Z' through cell c requires the following conditions: 1) $L(Z') > L(Z' \cup \{c\})$, which indicates that Z' did not expand to c, and 2) $L(Z'') > L(Z')$ and $L(Z'') > L(Z)$, which indicates that the merged cluster Z'' is more significant than either Z or Z'. It is easy to prove that it never happens when $L(Z)$ is monotonic with $\frac{n_Z}{N_Z}$, because when finding overdensity, if $\frac{n_{Z'}}{N_{Z'}} > \frac{n_{Z'}+n_c}{N_{Z'}+N_c}$, we have either $\frac{n_{Z'}+n_Z+n_c}{N_{Z'}+N_Z+N_c} < \frac{n_{Z'}}{N_{Z'}}$ or $\frac{n_{Z'}+n_Z+n_c}{N_{Z'}+N_Z+N_c} < \frac{n_Z}{N_Z}$. When finding underdensity, such a conclusion also holds with the directions of the inequations reversed. However since $L(Z)$ is neither monotonic with $\frac{n_Z}{N_Z}$ nor with N_Z, the algorithm needs to handle possible cluster merging.

An additional criterion for seed and growing cell selection is function AboveMinPOI (Algorithm 4), which is to restrict extra large size of a cluster. From (3.1) and (3.2) we can infer that if a cluster Z contains cells inside which no point is observed, the value of $\log L(Z)$ keeps unchanged. In GridScan, a parameter minPOI (default value 1) is used to limit the appearance of such "empty" cells. When the objective is to find overdensity, only cells with case number greater than minPOI will be considered as a candidate seed or growing cell. When the objective is to find underdensity, only cells with control number greater than minPOI will be considered.

3.3 Computational Complexity We first consider the time complexity of finding one most likely cluster in GridScan after the data is aggregated into $N \times N$ grids. It can be characterized by the time complexity of FindOneCluster (Algorithm 3). The dominant computation in lines 1–5 is finding seed (line 2) which costs $O(N^2)$. Then in the cluster growing (lines 6–18), primary computation is the search for c_{next} among neighbors (line 8), considering that AboveMinPOI only costs $O(1)$ and GetNeighbors also costs $O(1)$ if we always keep neighbors together with Z whenever returning Z from FindOneCluster. The rest set operations can all have efficient implementation that their time complexities are dominated by line 8. The time complexity of line 8 relies on the size of neighbors. Let a_i denote |neighbors| in the ith iteration, $i = 1, 2, \ldots, m$, and suppose in every iteration the average increment of |neighbors| is d, where by simply enumerating all possible cases we can find that $1 \le d \le 4$ under 8-connectedness. Approximately we can construct an arithmetic sequence: $a_1 = O(8), \ldots, a_i = a_{i-1} + d, \ldots, a_m = O(4N) = a_1 + (m-1)d$. Thus we have $m = \frac{O(4N)-O(8)}{d} + 1$, and the total time complexity of cluster growing is $\sum_{i=1}^{m} a_i = \frac{m}{2}(O(8) + O(4N)) = O(N^2)$.

Overall, in Algorithm 1, if we further include the data aggregation (line 2) and the Monte Carlo simulation (lines 5–9) which are also common steps in all grid-based approaches, the time complexity of GridScan is $O(K \cdot (n + N^2))$.

3.4 Discussion of Parameters GridScan only has two control parameters N and minPOI impacting the

cluster growing. Whereas K and θ are commonly used among all statistical hypothesis testings (not only in cluster detection context) with widely accepted settings (K=999, θ=0.05). K determines p-value precision, and θ reflects the unusualness level in which user is interested. They do not influence the cluster growing.

N defines the spatial resolution on which the cluster is detected. A larger N will lead to finer spatial scales of cluster detection, and vise versa. Cluster detection result can be different with varying N. This refers to the well-known modifiable areal unit problem (MAUP) [19] that both the scale effect and the zoning effect influence the result. Considering the grid aggregation in GridScan has been widely used in all grid-based approaches, e.g., [24, 22, 16, 18, 1], the MAUP in the context of cluster detection becomes a separate issue. A possible solution to MAUP can be to perform cluster detection at different scales and analyze the effects. This can be easily done in GridScan since it is efficient. In specific scenarios, the choice of a good N depends on the characteristic of input data. On the other hand, an appropriate N can be determined by the granularity in which the user is interested. It is reasonable to let the user interactively try different N's to view and understand the results on different levels of granularity. Although we can't give in-depth theoretical analysis of MAUP for GridScan in this paper, in experiments we will study the effect of varying N. Experiments can show that although results can be different, they are all reasonable and correct in terms of characterizing clusters at the defined scale and zoning.

As already mentioned, minPOI is used to exclude "empty" cells. A larger minPOI will lead to more conservative cluster growing, which will be shown in experiments. To our experience, minPOI =1 works well enough in most cases if there is no special requirement. One may doubt that this will lead to "over-fitting". However, we should notice that the cluster detection problem is different from the classification problem in machine learning field. No generalized prediction performance needs to be considered because the input dataset contains all available observations, not a set of samples, and thus there is no i.i.d. assumption. The essential of cluster detection is simply to characterize the true clusters as it is, and as accurately as possible. Therefore, "over-fitting" is not a defect but a required feature in such a context.

3.5 Extending to GridScan-Pro We adopt a similar method as used in [18, 3] to extend GridScan to a prospective spatio-temporal surveillance approach, namely GridScan-Pro. The spatio-temporal data to be analyzed is segmented into a sequence of chunks, i.e., temporally aggregated by a regular time interval. GridScan is used to compare a newly observed data chunk D_t with a historical chunk D_{t-1} to see if significant clusters can be detected at time frame t. In common cases, the amount of data evolves nearly continuously over time, thus we can assume that $|D_t|$ will not differ much with $|D_{t-1}|$. Therefore, we specifically use the Bernoulli model based density measure (3.1) in GridScan and treat data in D_{t-1} as controls and data in D_t as cases. Significant change of data points' distribution, e.g., emerging of a new cluster, expanding or moving of an existing cluster, can be detected by GridScan-Pro immediately when they appear. Although the framework of GridScan-Pro is simple, later in experiments we will demonstrate its promising performance. The main steps of GridScan-Pro is shown in Algorithm 5.

Algorithm 5: GridScan-Pro

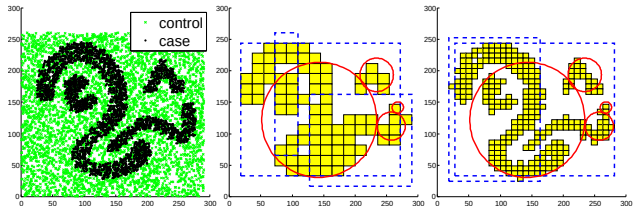
Input: Data chunks D_t, D_{t-1}; N; minPOI; K; θ
Output: A set of significant spatio-temporal clusters detected in D_t, STClusters

1 construct a temporary point dataset $D = D_t \cup D_{t-1}$, where points in D_{t-1} are labeled as controls and points in D_t are labeled as cases;
2 STClusters $\leftarrow$ `GridScan`$(D, N,$ minPOI$, K, \theta)$; `// using the Bernoulli model`
3 **return** STClusters;

4 Experiment

We will test GridScan (and -Pro) on 9 datasets. So far, to the best of our knowledge, there is no standard benchmark suite for point data overdensity/underdensity detection commonly used in prior arts, publicly available, and with known ground truth – although quite a few areal data datasets are available. Therefore, we generated 6 synthetic datasets and obtained 3 datasets from previous researchers, all of which have ground truth for fair evaluation. Kulldorff's spatial scan statistic using circular and elliptic windows [10, 13] and the space-time scan statistic using cylindrical window [11], Neill's scan statistic searching for rectangular regions [16], and PSVC [3] are involved for comparison. We use SaTScan software [14] as the implementation of Kulldorff's spatial and space-time scan statistic[2], and *city4_applic* software [17] which uses the Poisson model searching for overdensity as the implementation of Neill's scan statistic. GridScan is implemented using C++. Among all, only SaTScan performs Monte Carlo simulation with multi-thread support. Since no source code or software is available for PSVC, we can only compare with its results reported in [3] in spatio-temporal tests. Both GridScan and Neill's scan statistic determine $N \times N$ grids automatically from data. We let SaTScan run on

[2]The non-compactness penalty parameter is set to "medium" when using elliptic window in SaTScan.



(a) Dataset1 (b) $N = 16$ results (c) $N = 32$ results

Figure 2: Dataset1 and results. GridScan detects two clusters with N=16 and three with N=32 (yellow grids). SaTScan detects four clusters (red circles). Neill's scan statistic detects three overlapping clusters with N=16 and two with N=32 (rectangles with blue dashed lines).

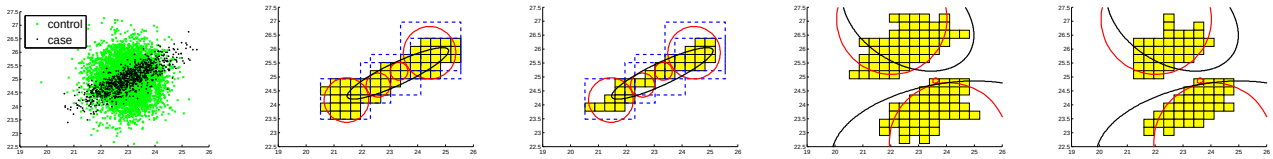
data points without spatial aggregation to verify the advantage of grid-based approaches in computational efficiency. If not specifically mentioned, we use default minPOI =1, K=999 and θ=0.05 for GridScan hereafter. All experiments are done on a Windows XP machine with 2.53GHz Intel Core 2 Duo CPU and 3G memory.

4.1 Purely Spatial Test We compare GridScan, Kulldorff's spatial scan statistic (denoted by SaTScan hereafter), and Neill's scan statistic on two spatial datasets. Dataset1 is generated based on a widely used dataset for testing traditional clustering algorithm which is able to find irregular shaped clusters, e.g., DBScan [7]. It contains 9776 points, among which 4888 case points constitute four local areas of overdensity with very irregular shapes. For testing the ability to accurately characterize irregularly shaped clusters, we neglect the significance of clusters, i.e., a cluster will always be reported despite of its p-value. But the p-value is still to be computed and included in overall CPU time. For GridScan and Neill's scan statistic, we use N=16, 32 to test their performance on two scales. SaTScan runs on points, so its results do not change with N. When using elliptic window, SaTScan returns an "insufficient memory" error, thus only results of circular window are reported. For GridScan and SaTScan, we choose to use the Bernoulli model since the amount of cases is comparable to the amount of controls. The *city4_applic* software only supports the Poisson model, but results can still be compared. The dataset and results are shown in Fig. 2. We can see that although the number of clusters is not correctly figured out by any of the approaches, obviously, GridScan performs the best in accurately detecting the extents and locations of the overdensities on both the two scales.

Dataset2 is used to test both overdensity and underdensity detection. It is generated as follows: 5000 control points are sampled from a Gaussian distribution with mean vector $(23, 25)$ and covariance matrix $\begin{pmatrix} 0.4 & 0 \\ 0 & 0.4 \end{pmatrix}$; 800 case points are sampled from a Gaussian distribution with the same mean but a different covariance matrix, $\begin{pmatrix} 0.7911 & 0.3632 \\ 0.3632 & 0.2440 \end{pmatrix}$. The true cluster of overdensity nearly covers the region where cases distribute. But since the density of controls is also high around the mean, an accurate characterization of the cluster should be like a dumbbell rather than a perfect ellipse. The true clusters of underdensity are two separate regions which contain mainly controls but few cases inside. We set N=16 for GridScan and Neill's scan statistic, and use the Poisson model for all approaches since cases are relatively fewer. We test minPOI =1, 3 in GridScan to see its impact. The dataset and detected clusters significant at 0.05 level are shown in Fig. 3.

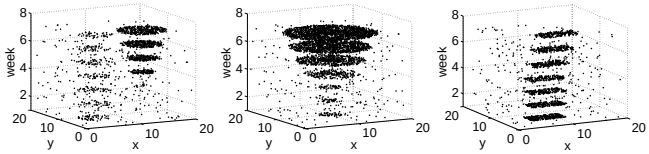
In overdensity test, GridScan with minPOI =1 accurately detects the true cluster and reports it as a whole. SaTScan using circular window almost successfully identifies the entire true cluster, but reports four separate circles. When using elliptic window, it can only identify the central area of the true cluster due to the shape restriction. Neill's scan statistic also almost covers entire region of the true overdensity, but reports five overlapping rectangles. It fails to accurately characterize the dumbbell-like shape by including extra large region in the central area. In underdensity test, Neill's scan statistic is not included since the *city4_applic* software only supports searching overdensity. SaTScan reports three circles or two ellipses which obviously include unnecessarily large regions. Again, GridScan with minPOI =1 detects the true clusters more accurately. We can also find that when minPOI becomes larger (e.g., 3 in our tests), the clusters detected by GridScan will shrink because less cells are qualified for growing. But the results are also reasonable and correct considering the stricter growing criterion.

4.2 Spatio-Temporal Surveillance Test Three spatio-temporal datasets, "emerging", "expanding" and "moving" are adopted from the author of [3] for surveillance tests. The datasets are shown in Fig. 4. The data points' coordinates are spatially distributed in $[0, 20]^2$ and temporally aggregated into 7 weeks. Similar datasets were used to test PSVC [3]. The objective is to detect overdensities implying significant changes of data points' distributions. In the "emerging" dataset, a new cluster emerges away from a permanent cluster since week 3 and spreads over time. However, only 15 points belong to the newly emerging cluster are observed in week 3. In week 4 the anomaly becomes clearer. In the "expanding" dataset, an existing circular cluster begins to expand since week 4. The true anomaly is a large scale ring shape. In the "moving" dataset, a rectangular cluster begins to move since week 5. The true



(a) Dataset2 (b) Overdensity results, minPOI=1 (c) Overdensity results, minPOI=3 (d) Underdensity results, minPOI=1 (e) Underdensity results, minPOI=3

Figure 3: Dataset2 and results significant at 0.05 level. GridScan results (all have p-value 0.001) with minPOI=1, 3 are shown by yellow grids. SaTScan results are shown by red circles and black ellipses. Neill's scan statistic results are shown by rectangles with blue dashed lines (in overdensity test only).



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

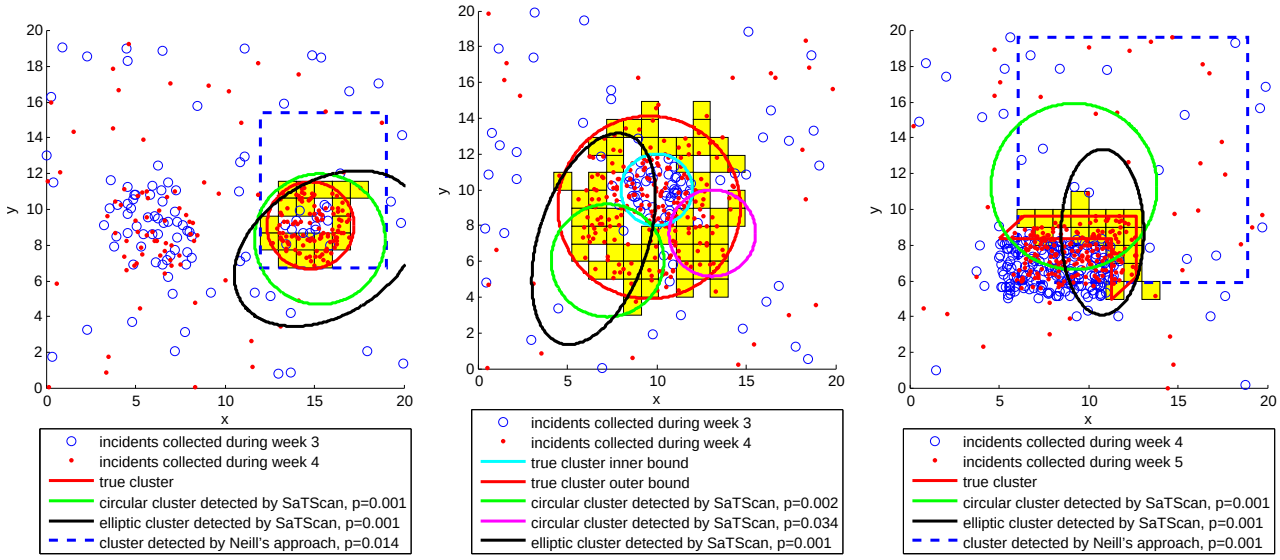
Figure 4: 3D view of the 3 spatio-temporal datasets.

anomaly is an L-shaped region. In all the datasets, a small amount of uniformly distributed noise also exists.

Using the framework of GridScan-Pro, we can substitute the GridScan component (line 2 in Algorithm 5) with Neill's scan statistic and SaTScan (the spatial scan statistic using circular and elliptic windows). The resulting approaches can be fairly compared with GridScan-Pro under the same methodology to deal with spatio-temporal data. In the tests, we use $N = 20$ for GridScan-Pro and Neill's scan statistic. Fig. 5 shows the results at the time (week) that the clusters are detected. On the "emerging" dataset, all approaches detect the change in week 4. Compared with the results of SaTScan and Neill's scan statistic, GridScan-Pro characterizes the cluster much closer to the ground truth. On the "expanding" dataset, both GridScan-Pro and SaTScan can detect the change in week 4, but GridScan-Pro detects the true anomaly more accurately. However, Neill's scan statistic fails to detect any significant cluster in week 4. On the "moving" dataset, all approaches detect the change in week 5. GridScan-Pro almost perfectly detects the true anomaly while others perform much worse.

PSVC and the space-time scan statistic using cylindric window [11] have been tested on three similar datasets [3]. The datasets are not exactly the same as what we have and also they are unavailable. However, they demonstrate similar scenarios. Due to the page limit, PSVC results in [3] are not shown here. But as reported in [3], PSVC and the space-time scan statistic can detect the anomalies at about the same time. Based on the similarity of the data, we can infer that GridScan-Pro, PSVC, and SaTScan have approximately the same performance in terms of detecting significant changes at an early stage. Since all these approaches are based on the Bernoulli model, we may infer that the failure of Neill's scan statistic on the "expanding" dataset is partly due to the characteristic of the Poisson model. We can also find that GridScan-Pro exhibits similar performance to PSVC in accurately characterizing irregularly shaped clusters, which is obviously advantageous over the others. But as mentioned in Section 1, PSVC's parameters significantly influence its result and are hard to determine, and its overall time complexity is $O(Kn^2)$ given dataset size n. The advantages of GridScan-Pro over PSVC include: 1) setting parameters for GridScan-Pro is easier since the two parameters have clear physical implications, and 2) GridScan-Pro's overall time complexity $O(K \cdot (n + N^2))$ is much lower when applied to large datasets where usually $N^2 \ll n$.

4.3 Quantitative Analysis and Discussion We summarized the maximum $\log L(Z)$ obtained by the approaches and their CPU time in above experiments in Table 1. Although the $\log L(Z)$ of Neill's scan statistic on Dataset1 is not directly comparable with others because of different probabilistic model, in general, GridScan(-Pro) always finds the largest maximum $\log L(Z)$ in all overdensity tests, which indicates the advantage of GridScan(-Pro) when the true cluster is irregularly shaped. In the underdensity test, GridScan finds smaller maximum $\log L(Z)$ than SaTScan. This is because SaTScan does not restrict the size of circle/ellipse, and the specific control data distribution in Dataset2 enables the extra large clusters detected by SaTScan to contain more controls while not including any case, which enlarges the $\log L(Z)$. GridScan stops growing cluster at those cells based on minPOI criterion. Combined with human interpretation, GridScan result is more compact, accurate and reasonable. Given fixed minPOI, larger N could lead to larger maximum $\log L(Z)$ because the details of a true irregularly shaped cluster can be more accurately characterized. But a smaller N may also lead to multiple true clusters de-



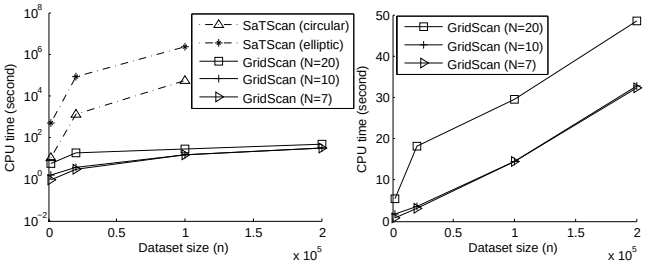
(a) "emerging" cluster detected in week 4 (b) "expanding" cluster(s) detected in week 4 (c) "moving" cluster detected in week 5

Figure 5: Results on the 3 spatio-temporal datasets. The true clusters from which the data points are generated are also explicitly shown. In each data chunk, GridScan-Pro detects one cluster with p-value 0.001 (yellow grids with black border). On the "expanding" dataset, Neill's scan statistic can't find a significant cluster at 0.05 level in week 4.

tected as a whole due to coarse granularity, and thus also enlarges maximum $\log L(Z)$ depending on the number of cases and controls in the cluster. This is what we can see from Dataset1 result. Given fixed N, a larger minPOI means stricter condition when evaluating a candidate cell for growing, thus will result in smaller maximum $\log L(Z)$.

In terms of CPU time, GridScan(-Pro) is fast in general. Although Neill's scan statistic is faster on Dataset1, it can't characterize the true clusters as accurately as GridScan can. SaTScan using circular window runs much faster than using elliptic window. But compared with grid-based approaches, running on data without spatial aggregation is much slower, even with multi-thread support. Larger N will surely increase the CPU time cost of GridScan when fixing minPOI. Given fixed N, larger minPOI can reduce the CPU time because the cluster growing stops earlier. Generally, considering the quality of results, GridScan's computational cost is low.

4.4 Scalability Test Four datasets are generated to test the scalability of GridScan: D1K with 10^3 cases and 10^3 controls; D10K with 10^4 cases and 10^4 controls; D50K with 5×10^4 cases and 5×10^4 controls; D100K with 10^5 cases and 10^5 controls. The scalability of GridScan-Pro is decided by GridScan, thus only GridScan is tested. SaTScan using circular and elliptic windows are also tested for comparison. We set N =7, 10, 20 for GridScan and using the Bernoulli model. Fig. 6 shows the CPU time against the input dataset size n. Fig. 7 shows the CPU time against the number of grids N^2. Because SaTScan fails to run on D100K due to out-of-memory error, and SaTScan using elliptic window can not finish in acceptable time on D50K, these results are not shown.



(a) (b)

Figure 6: CPU time against n. (a) Comparison with SaTScan (log scaled vertical axe). (b) GridScan only (linear scaled vertical axe).

We can see from Fig. 6 that the computational time of SaTScan grows fast as dataset size n grows because of $O(K \cdot n^2)$ overall complexity. We can also expect that the SVM based approach has similar scalability since they are of the same order. On the other hand, the scalability of GridScan is much better that with fixed

Table 1: Comparisons of maximum $\log L(Z)$ and CPU time (shown in bracket). The largest maximum $\log L(Z)$ in each test is shown in bold.

	GridScan(-Pro)	Neill's scan statistic	SaTScan: circular window	SaTScan: elliptic window
Dataset1	N=16: **1583.69** (8.343s) N=32: **857.262** (49.86s)	N=16: 539.569 (1s) N=32: 615.514 (5s)	392.750 (5m38s)	insufficient memory error
Dataset2 Overdensity	minPOI =1: **242.883** (2.781s) minPOI =3: **200.109** (1.516s)	85.141 (17s)	61.042 (1m54s)	142.031 (1h19m6s)
Dataset2 Underdensity	minPOI =1: 113.286 (3.078s) minPOI =3: 109.710 (2.875s)	–	126.615 (1m44s)	**137.059** (1h20m39s)
"emerging"	**36.3827** (3.766s)	9.55162 (166s)	30.7689 (1s)	31.4163 (37s)
"expanding"	**49.6349** (4.328s)	not detected (3s)	12.2105 (1s)	17.0193 (54s)
"moving"	**33.9758** (3.328s)	15.6311 (158s)	26.8212 (1s)	28.4956 (1m39s)

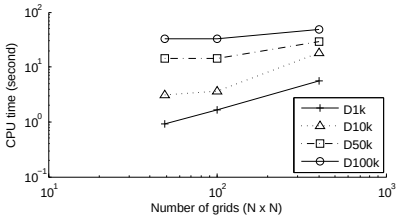


Figure 7: CPU time against N^2.

N^2, its CPU time grows linearly with n. From Fig. 7, we can find that with fixed n, its CPU time grows also linearly with N^2. There is only a little more overhead when n or N^2 becomes large. When dataset is very large, i.e., $N^2 \ll n$, the time complexity of GridScan will be dominated by $O(Kn)$. Therefore on larger datasets, the time increment trend in Fig. 7 becomes flat as N^2 grows. These results verify the analysis in Section 3.3, and show the good scalability of GridScan.

5 Case Study

A case study of characterization of epidemic spread is conducted. The dataset is from the IEEE VAST Challenge 2011 Mini-Challenge 1[3], including a large set of anonymized microblog messages (totally 1023077 messages) collected from computers and cellphones in a city from Apr 30 to May 20. Each microblog has GPS coordinates and time labels. We will apply GridScan to analyze the microblogs, detect when and where (ground zero location) the outbreak of an epidemic started, and characterize the pattern of the epidemic spread. The city map is shown in Fig. 8.

In our analysis, the microblog messages are first preprocessed by a text analyzing tool, MALLET [15], with Latent Dirichlet Allocation method to extract epidemic keywords including fever, chill, diarrhea, stomach, etc., to tag each microblog. Two keywords among all, fever and diarrhea, are chosen for further analysis, since they reflect two different representative symptoms. We applied purely spatial GridScan on daily collected microblogs, to detect where and when people talked about fever and diarrhea with significant overdensity, which could offer clue to detect the epidemic outbreak and spread pattern although noise may also exist.

A priori knowledge is that most people working in the city center in daytime, and the distribution of city population changes between daytime and night due to commuting during work hours. Therefore, the daily data is further split into daytime (8am to 6pm) and night (6pm to next day 8am) parts. Here we deal with spatio-temporal data in a different manner as in GridScan-Pro. The data is temporally aggregated by daytime and night into chunks as usual. But in each chunk, microblogs are divided into cases and controls. For fever analysis, microblogs tagged by fever are regarded as cases, the rest are regarded as controls. For diarrhea analysis, similarly, microblogs tagged by diarrhea are regarded as cases, the rest are regarded as controls. GridScan is applied twice on each chunk, one for fever and the other for diarrhea. Since here we want to detect cluster by comparing cases and controls within every chunk, but not detect change over multiple chunks, GridScan is used instead of GridScan-Pro.

We approximately treat the microblogs' Lat. Long. coordinates as plane coordinates. We set the regular grid cell size to $0.01° \times 0.01°$ for GridScan, and thus construct 15×38 grids, using the upper left corner point of the map (42.3017N, 93.5673W) as the upper left origin of grids. The Poisson model is adopted considering cases are relatively much fewer than controls. The detection results on each chunk are summarized in Table 2.

No cluster significant at 0.05 level can be detected for either fever or diarrhea until May 18. In daytime of May 18 and May 19, a significant cluster (p-value=0.001) of microblogs mentioning fever is detected covering central and eastern areas of the city (see Fig. 9). Although the number of "fever" cases grows to its peak value at May 18 night as can be seen from Table 2, because people getting back home from city center neutralize the overdensity, the significance of clusters decreases. Similar situation happens at May 19 night. Considering the noisy nature of microblog data, cluster with p-value

[3] http://hcil.cs.umd.edu/localphp/hcil/vast11/index.php/taskdesc/index

Table 2: Results on VAST Challenge 2011 epidemic dataset. For each chunk, three kind of information is summarized. First, numbers of cases, controls, and population in the chunk. Second, number of all candidate clusters detected by GridScan despite their p-values, and CPU time of GridScan. Third, log(L), p-value, and size (number of marked grids) of the most significant cluster detected by GridScan.

Fever	#case	#control	#pop	#cluster	CPU time	log(L)	p-value	size
May18day (May 18 8am – 6pm)	739	25470	26209	1	8.985s	165.598	0.001	69
May18night (May 18 6pm – May 19 8am)	966	39719	40685	2	13.234s	64.1115	0.121	43
May19day (May 19 8am – 6pm)	613	29614	30227	1	9.547s	160.343	0.001	75
May19night (May 19 6pm – May 20 8am)	557	42851	43408	4	11.375s	58.6143	0.023	44
May20day (May 20 8am – 6pm)	217	27174	27391	10	7s	20.5745	0.028	2
May20night (May 20 6pm –)	159	20605	20764	6	5.828s	18.1357	0.027	10
Diarrhea	#case	#control	#pop	#cluster	CPU time	log(L)	p-value	size
May18night (May 18 6pm – May 19 8am)	96*	40589	40685	1	7.609s	248.619	0.001	44
May19day (May 19 8am – 6pm)	149	30078	30227	1	6.516s	331.344	0.001	46
May19night (May 19 6pm – May 20 8am)	414	42994	43408	1	10.547s	842.7	0.001	61
May20day (May 20 8am – 6pm)	430	26961	27391	1	8.625s	807.928	0.001	71
May20night (May 20 6pm –)	247	20517	20764	1	6.516s	530.437	0.001	56

*All these cases are observed after May 19 0am.

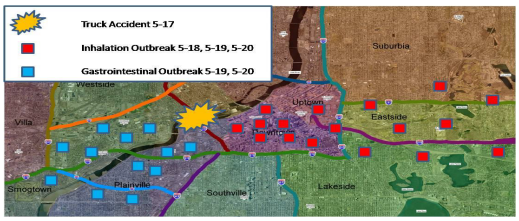


Figure 8: City map and illustration of the ground truth of the VAST Challenge 2011 epidemic dataset. The truck accident happened on a bridge over the river, which is the ground zero location.

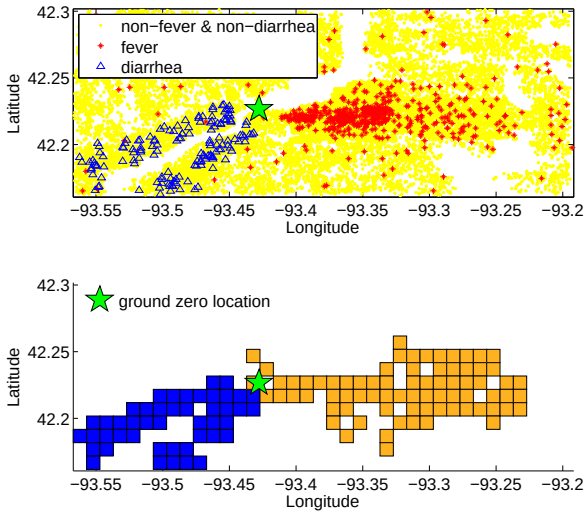


Figure 9: Data and clusters detected by GridScan in May19day chunk. Fever cluster is shown in orange, whose location is similar to the one detected in May18day chunk but with larger extent to the east. Diarrhea cluster is shown in blue. Diarrhea clusters detected in other chunks look similar to this one.

larger than 0.01 may no longer effectively reflect the epidemic outbreak and spread. However, the two clusters detected in May 18 and May 19 daytime do indicate something unusual, i.e., an epidemic outbreak. In May 20 day and night, GridScan finds a lot more small local clusters, but their p-values are relatively large, too. For microblogs mentioning diarrhea, a significant cluster (p-value=0.001) emerges and persistently locates around the downstream of a river across the city since May 18 night, more precisely, during May 19 0am to 8am. The shape, extent and location of the cluster do not change much as time goes. Visualization of the GridScan results in May 19 daytime chunk is shown in Fig. 9. Interestingly, the fever cluster touches the diarrhea cluster around a point (42.22655N, 93.42752W), which is the location of a bridge over the river. This implies that if the two disease clusters are caused by a same reason, then that bridge is likely to be the ground zero location.

The ground truth is also shown in Fig. 8 that a truck accident happened on that bridge at May 17 noon, and the truck was carrying a large amount of food contaminated by harmful spores. Due to the accident, the spores were dispersed into the air, and also fell into the river which is used for obtaining drinking water. The spores spread by west wind and water current in the following days, which caused the epidemic outbreak. Therefore, people got infected by two kinds of transmissions, airborne and waterborne. Due to different incubation period and spread speeds, the true airborne disease outbreak originated near the bridge on May 18 and dispersed over a large cone-shaped area in the direction of the west wind. The true waterborne disease outbreak also originated near the bridge on May 19 and dispersed along the river in the direction of the water current which flows Southwest. Some people with the airborne disease went to hospitals around the city in May 20, which leads to a larger number of small clusters detected by GridScan, although they are not that significant. But people with waterborne disease rarely moved, which makes the diarrhea cluster persist.

We can find that GridScan detects the outbreak in time and correctly characterizes the spread patterns for both the airborne and waterborne diseases with low computational cost. It also gives useful clue for identifying the true ground zero location.

6 Conclusion

In this paper, we propose a grid-based cluster detection approach GridScan for detecting irregularly shaped significant clusters for point data. GridScan is efficient that its time complexity is just $O(N^2)$ in finding the most likely cluster given $N \times N$ grids, in spite of dataset size n. Its overall time complexity is linear to $O(n + N^2)$. A prospective spatio-temporal cluster detection approach GridScan-Pro is also proposed for data change surveillance by extending GridScan. Although GridScan employs a local and greedy search strategy, experiments and a case study have shown that it significantly outperforms existing approaches in accurately characterizing the true clusters on various datasets. GridScan-Pro also shows its effectiveness of detecting significant data changing patterns with irregular shape immediately when they appear. Compared with the SVM based approach whose performance is also promising, the parameters of GridScan are easier to set. More importantly, GridScan's efficiency and scalability are much better, which makes it suitable to analyze very large spatial and spatio-temporal datasets.

References

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

[2] D. Birant and A. Kut. ST-DBSCAN: An algorithm for clustering spatial-temporal data. *Data & Knowledge Engineering*, 60(1):208–221, 2007.

[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] J. Conley, M. Gahegan, and J. Macgill. A genetic approach to detecting clusters in point data sets. *Geographical Analysis*, 37(3):286–314, 2005.

[5] J. Devine and A. Stefanidis. A support vector clustering based approach for spatiotemporal analysis in security informatics. In *Int Arch Photogram Rem Sens Spatial Inform Sci, vol. XXXVII, part B2*, pages 1–6, 2008.

[6] L. Duczmal, A.L.F. Cancado, R.H.C. Takahashi, and L.F. Bessegato. A genetic algorithm for irregularly shaped spatial scan statistics. *Comput. Statist. Data Anal.*, 52(1):43–52, 2007.

[7] M. Ester, H.P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD'96*, pages 226–231, 1996.

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

[9] V.P. Janeja and V. Atluri. Random walks to identify anomalous free-form spatial scan windows. *IEEE Transactions on Knowledge and Data Engineering*, 20(10):1378–1392, 2008.

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

[11] M. Kulldorff. Prospective time periodic geographical disease surveillance using a scan statistic. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 164(1):61–72, 2001.

[12] M. Kulldorff, WF Athas, EJ Feurer, BA Miller, and CR Key. Evaluating cluster alarms: a space-time scan statistic and brain cancer in los alamos, new mexico. *Am. J. Pub. Health*, 88(9):1377–1380, 1998.

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

[14] 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`.

[15] A.K. McCallum. MALLET: A machine learning for language toolkit, 2002. `http://mallet.cs.umass.edu`.

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

[17] D.B. Neill, A.W. 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`.

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

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

[20] G.P. Patil and C. Taillie. Upper level set scan statistic for detecting arbitrarily shaped hotspots. *Environmental and Ecological Statistics*, 11(2):183–197, 2004.

[21] R. Sahajpal, G.V. Ramaraju, and V. Bhatt. Applying niching genetic algorithms for multiple cluster discovery in spatial analysis. In *International Conference on Intelligent Sensing and Information Processing*, pages 35–40. IEEE, 2004.

[22] G. Sheikholeslami, S. Chatterjee, and A. Zhang. Wavecluster: A multi-resolution clustering approach for very large spatial databases. In *VLDB'98*, pages 428–439, 1998.

[23] K. Takahashi, M. Kulldorff, T. Tango, and K. Yih. A flexibly shaped space-time scan statistic for disease outbreak detection and monitoring. *International Journal of Health Geographics*, 7(1):14, 2008.

[24] W. Wang, J. Yang, and R. Muntz. STING: A statistical information grid approach to spatial data mining. In *VLDB'97*, pages 186–195, 1997.