A LRT Framework for Fast Spatial Anomaly Detection

Mingxi Wu,[1] Xiuyao Song,[2] Chris Jermaine,[3,4] Sanjay Ranka,[4] and John Gums[4]
[1]Oracle Corporation, Redwood Shores, CA 94065
[2]Yahoo!, Inc., Sunnyvale, CA 94089
[3]Rice University, Houston, TX 77251
[4]University of Florida, Gainesville, FL 32611
mingxi.wu@oracle.com, {xsong,cjermain,ranka}@cise.ufl.edu, jgums@ufl.edu

ABSTRACT

Given a spatial data set placed on an $n \times n$ grid, our goal is to find the rectangular regions within which subsets of the data set exhibit anomalous behavior. We develop algorithms that, given any user-supplied arbitrary likelihood function, conduct a likelihood ratio hypothesis test (LRT) over each rectangular region in the grid, rank all of the rectangles based on the computed LRT statistics, and return the top few most interesting rectangles. To speed this process, we develop methods to prune rectangles without computing their associated LRT statistics.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database application—*data mining, spatial databases and GIS*

General Terms

Algorithms, Experimentation

1. INTRODUCTION

Discovering subsets of database data that are spatially close to one another and exhibit anomalous behavior is of key importance in many application areas. For example, consider our motivating application of mining antimicrobial (antibiotic) resistance patterns. Antimicrobial resistance in nosocomial (hospital acquired) bacterial infections is a key public health problem. Antimicrobial drugs are the first and sometimes only means of attacking bacterial infection, but due to use and misuse over time, antimicrobials become less useful as bugs become resistant due to selective pressures. The result is that common, often mild, nosocomial infections such as staph can become deadly with no effective treatment. The Antimicrobial Resistance Management (ARM) database (http://www.armprogram.com) consists of antimicrobial resistance data for nearly 400 hospitals over a 15-year period, and presents an opportunity to study the epidemiology of antimicrobial resistance. Over those 15 years, the trend in resistance rates is generally upward. However, a key question that we would like to answer is: Is the trend *uniformly* upward over time, or are there spatial regions where a set of hospitals have significantly different trends? Knowing the answer would provide key insight into the epidemiology of antimicrobial resistance, indicating, for example, how "mobile" the bugs are, or how the evolution of bugs in a hospital's general region affects local resistance rates. If resistance trends show a strong geographic affinity, then it might indicate that resistance is a local phenomenon, and so local programs at individual hospitals aimed at careful antimicrobial stewardship could be useful. However, if resistance trends are uniform over a wide area, then it might indicate that antimicrobial stewardship must be a wider effort.

Applying a Spatial Likelihood Ratio Test. Using classical statistics, we can mine the ARM database for regions of the country with anomalous or unique resistance trends. Given a spatial region, we might treat the number of resistant cases in year y as the result of a single binomial trial, with an unknown probability of resistance p_y and the number of experimented isolations of the bug as a known, fixed input n_y. Since we are interested in trends, we could link all of the p_y values over time for a given region using a linear model over the years, $p_y = y\Delta + p_0$. Then, using a likelihood ratio test [12], we could check whether the trend Δ for the given spatial region is significantly different from the trend that would be used for the entire country. By breaking the country into a grid and checking each contiguous region in the grid for a difference in the local trend, we will locate any locally anomalous resistance trends.

So, what's the problem? As long as checking whether each contiguous region is anomalous is computationally inexpensive, then a brute-force search is feasible. There are $O(n^4)$ rectangular regions in an $n \times n$ grid. For example, if $n = 32$, then there are 278,784 regions. This is not too many regions, but the overall cost to search the grid using a brute-force method is $O(cn^4)$, where c is the average cost to check a given area by computing a single likelihood ratio test. The likelihood ratio test resulting from trend-based search described above may require seconds to run for each region that is searched, requiring weeks to run on a 32 by 32 grid. Therefore, the problem is not enumerating all of the local regions to check; *the problem is actually having to run the likelihood ratio test on all of them*.

Our Contributions. The primary contribution in this paper is to generalize the set of statistical models that can be used for this sort of spatial search. We propose using the classic likelihood ratio test (LRT) statistic as a score function to evaluate the "anomalousness" of a given spatial region with respect to the rest spatial data. The LRT is quite general: it works with virtually any underlying statistical model. A user of our framework need only supply implementations of a few specific functions to instantiate our framework. But a key problem in practice is that computing even one test statistic value can be very expensive. Thus, we propose a pruning strategy

KDD'09, June 28–July 1, 2009, Paris, France.

that works for almost any underlying likelihood function, that can be used to radically cut down on the number of likelihood ratio tests that must be run when searching for anomalous spatial regions.

2. BACKGROUND

2.1 The LRT Statistic

The LRT is a hypothesis test that facilitates the comparison of two models: one parametric stochastic model associated with the hypothesis that there is an anomaly, and another associated with the hypothesis that there is no anomaly—the so-called "null model". Both parametric models are embodied by identical likelihood functions $L(\theta|X)$, where X contains the values output by the underlying stochastic process, and θ is a set of parameters coming from the parameter space Θ. The (restricted) parameter space Θ_0 allowed under the null model is the complement of the parameter space allowed in the case of anomalous data, denoted as $\Theta - \Theta_0$. To check for an anomaly using the LRT, two hypotheses $H_0 : \theta \in \Theta_0$ and $H_a : \theta \in \Theta - \Theta_0$ are compared by computing the statistic:

$$\lambda(X) = \frac{\sup_{\Theta_0} L(\theta|X)}{\sup_{\Theta} L(\theta|X)} \quad (1)$$

This statistic is computed by first computing a maximum likelihood estimate (MLE) under both parameter spaces Θ_0 and Θ, and then computing the ratio of the likelihoods obtained via the two MLEs. Wilks [12] showed that the asymptotic distribution of $\Lambda(X) = -2\log\lambda$ (which we subsequently refer to as the *LRT statistic*) is chi-squared with $(p - q)$ degrees of freedom under the null hypothesis that $\theta \in \Theta_0$. p is the number of dimensions (or free parameters) in Θ, and q is the number of dimensions in Θ_0. Thus, to check for an anomaly at confidence level α, one checks whether $\Lambda(X) \geq c$, where c is a non-negative number computed by finding how far out in the tail of a chi-square distribution one has to go to find $(1 - \alpha)\%$ of the mass.

For a very simple example of the sort of case where the LRT is applicable, imagine that we wish to test whether the disease rate within a spatial area A is different than the disease rate outside of the area. We assume that the underlying stochastic process that generates the number of cases of disease is binomial: each person who lives in A has a certain, unknown probability of becoming ill in a given time period, and we wish to check whether this probability is different in A than it is outside of A. For each A, $X = \{k_A\}$ where k_A is the number of observed diseased individuals inside of A. The set of model parameters θ contains an unknown probability or rate of infection p_A, and the known number of individuals n_A who live inside of A. For a given A, if the null hypothesis holds and $\theta \in \Theta_0$, then $p_A = p_{\bar{A}}$ and the disease rates are the same within and without the given spatial area.

The likelihood function $L()$ would then be a binomial function:

$$L(\theta|X) \propto p_A^{k_A}(1 - p_A)^{n_A - k_A} p_{\bar{A}}^{k_{\bar{A}}}(1 - p_{\bar{A}})^{n_{\bar{A}} - k_{\bar{A}}}$$

The degrees of freedom of the null distribution is one, since there is one more free parameter allowed in Θ than in Θ_0.

One can easily use the LRT as a basis for spatial anomaly search. First, all contiguous, rectangular areas in a grid are searched, and the value of the LRT statistic is computed over each of them. Those areas with the greatest value for the statistic are returned to the user for further examination as potential anomalous areas.

2.2 The Spatial Scan Statistic

The LRT has been used before for spatial anomaly detection. For example, the LRT test forms the basis for the spatial scan statistic (SSS), which is useful for detecting a cluster of event occurrences in a spatial area. The SSS was first proposed in the statistics literature [6]. The model underlying the SSS assumes that each subregion has a Poisson process controlling the number of event occurrences within it. SSS-related work in the KDD literature has focused on searching for an anomalous area in an efficient fashion [9, 8, 1].

As we will discuss subsequently, our framework is similar to the SSS in that it utilizes the LRT to perform spatial anomaly detection, but it is far more general, admitting a very wide range of stochastic models, and so work on speeding detection via SSS is not obviously relevant. For example, existing SSS algorithms generally make extensive use of the fact that adding more data to an area while keeping the ratio of the number of observed events to the area measure constant must cause the "interestingness" of the area to increase. This is true in the case of a Poisson model, but not in the sort of general model that we consider. Thus, new ways to speeding up the computation are required.

3. PROPOSED LRT FRAMEWORK

We begin by assuming that the spatial area over which we search for anomalies has been pre-partitioned into an $n \times n$ spatial grid. For each rectangular area A in the grid, we wish to answer the question, "Does A differ significantly from the remainder of the area in the grid?" This question is answered by computing a LRT statistic that compares A to $\bar{A}$. If the value of the LRT statistic is large, then A is returned to the user as an anomalous region.

Our framework can be thought of as a generic template, which requires that a user supplies a few functions which instantiate a particular anomaly detection problem. We now describe the process that a typical user would undertake in order to apply our framework to a particular problem.

3.1 Choosing a PDF

The first thing that a user must do is to postulate an appropriate stochastic model to describe the data generation process within each cell. A cell is a minimum spatial unit within which we assume that there is no spatial variation, and so it does not make sense to subdivide a cell spatially when performing anomaly detection. The model used on a per-cell basis may be simple, such as a classical Poisson or Bernoulli model, or it may be an arbitrary, user-defined model of significant complexity. The stochastic model for a cell c is characterized by a probability density function (PDF), denoted by $f(X_c|\theta_c)$. The likelihood of a model given the data is then $L(\theta_c|X_c) = f(X_c|\theta_c)$. In order to make use of the LRT statistic, we must be able to calculate the likelihood of the entire grid. Assuming the generative processes within each cell are independent of each other[1] , the likelihood of the entire grid is given by:

$$L^* = \prod_{c \in Grid} L(\theta_c|X_c) \quad (2)$$

3.2 Defining the Test Set

Once the likelihood function has been defined, the second step in employing the LRT framework is specifying two competing hypotheses in such a way that LRT can be used to decide whether a testing rectangle is a spatial anomaly. Informally, given a test area A, the two competing hypotheses take the form:

[1]The cell independence assumption is made to keep the scope of the paper manageable. In practice, this assumption may not be too restrictive. For example, the Poisson model underlying the SSS assumes independence.

- H_0 : the process generating the data in the cells of A is not substantially different from the process generating the data in the cells outside of A.

- H_a : the process generating the data in the cells within A *is* substantially different from the process outside of A.

Since the generative process is modeled via a PDF that is assumed to be the same across all cells, determining whether the generative process differs from cell-to-cell is equivalent to determining whether the parameters to the process differ within and without A.

Note the word "substantially" in the definition of the two hypotheses. When we are trying to test whether the generative process in two different cells (or two groups of cells) differs, we are not interested in comparing every aspect of the generative process (or every parameter). There will be natural spatial variations in the generative process that we want to ignore. For example, when we are testing whether the trend of antimicrobial resistance is the same inside of an area A as it is outside of A, differences in the starting point of the trend are uninteresting.

As a result, our framework differentiates two types of parameters for a cell's PDF: the "shared parameters" and the "local parameters". The set of "shared parameters" is the subset of θ that is forced to be identical among each member of a group of cells. The shared parameters are used to model within-group cell-similarity, and it is from within the set of shared parameters where we find the particular data property or properties that are indicative of an anomaly. Those shared parameters that the user is actually interested in testing to see whether they are the same within a region A and outside of the region A are called the "test set", and are denoted by T. The test set is a non-empty subset of the shared parameters.

The "local parameters" are those parameters within θ that are customizable to each cell. They generally capture the uninteresting or anticipated spatial variation of the data across different cells. Sometimes, the precise values for local parameters may be known and supplied beforehand by the user (such as the number of people living in a spatial region). Other times, local parameters may be unknown before the test is run, and their values are inferred (such as the initial resistance rate when checking for different trends in antimicrobial resistance rates).

Example. Recall that in our motivating application, we want to find a spatial area where the trend in antimicrobial resistance is different inside of the area than it is outside of the area. We assume that the observed number of resistant cases in a cell is generated via a sample from a binomial random variable, where the number of microbes observed in year y is n_y, the probability of resistance in a given year y is p_y, and the linear function $p_y = \Delta \times y + p_0$ is used to model the trend of antimicrobial resistance rate over many years. In this case, for a given cell c, $\theta_c = \{\Delta, p_0, n_0, n_1, n_2 ...\}$. When detecting anomalous regions, we want to know whether there is an area A where the rate of change in resistance over time differs within A and outside of A. In this case, the rate of change Δ is a shared parameter that is assumed to be uniform inside of A, and uniform outside of A. The question is whether Δ has a single value for the entire grid. Thus, the test set $T = \{\Delta\}$. In contrast, p_0 is a local parameter that allows the trend to have a different starting point in each cell. p_0 for each cell is unknown and must be inferred from the data. The number of microbes n_y observed in year y is also a local parameter, but it is available to the testing framework and need not be inferred.

In this case, the two hypotheses that we are comparing become:

- H_0: $\Delta_A = \Delta_{\bar{A}}$

1. **For** each rectangle A in the grid
2. **Let** $\theta_G = MLE_0(f(G))$
3. **Let** $(\theta_A, \theta_{\bar{A}}) = MLE_1(f(G), A)$
4. **Let** $\Lambda = -2 \log L(\theta_G|X_G) + 2 \log L(\theta_A|X_A) + 2 \log L(\theta_{\bar{A}}|X_{\bar{A}})$
5. **If** Λ is in the top k found so far, then remember A

Figure 1: Naive top-k LRT search (Algorithm 1)

- H_a: $\Delta_A \neq \Delta_{\bar{A}}$

where Δ_A denotes the shared Δ within the current rectangle A and $\Delta_{\bar{A}}$ denotes the shared Δ outside of the rectangle A.

In general, the framework considers the following two competing hypotheses for each rectangular area A:

- $H_0 : \forall t \in T, t_A = t_{\bar{A}}$

- H_a: $\exists t \in T$ where $t_A \neq t_{\bar{A}}$

3.3 Instantiating the Framework

Once the user has developed a likelihood function and determined what parameters are in the test set, he or she must implement four functions in order for our framework to be used. These functions are as follows:

- The **summarizing function** $f(A)$ that returns the set of summary data X_A for a region A in the grid, as well as all local parameters whose values are known constants.

- A **likelihood function** $L(\theta_c|X_c)$ that accepts a set of cell summary statistics as well as a set of cell parameter values, and returns the associated likelihood.

- The **null-space MLE procedure** $MLE_0(f(A))$ that performs MLE in the null parameter space for an area A. That is, this MLE procedure accepts the output of $f(A)$: a set of summary statistics X_A over some spatial region A, as well as any local parameters for cells in A whose values are known constants. It then computes the set of non-constant parameter values in θ_A that maximize the likelihood function $L(\theta_A|X_A)$. This maximization is done under the constraint that for every two cells c_1 and c_2 in A, and for every shared parameter s, $s_{c_1} = s_{c_2}$.

- The **complete-space MLE procedure** $MLE_1(A, f(G))$ that accepts a region A as well as all data and constant-valued local parameters from the entire spatial grid G, and then chooses the non-constant values in θ_A and $\theta_{\bar{A}}$ that maximize the entire grid's likelihood: $L(\theta_A|X_A) \times L(\theta_{\bar{A}}|X_{\bar{A}})$. This is done under the constraint that all shared parameters not in the test set must have the same value for each and every cell. However, parameters in the test set T may differ inside of A and outside of A. For $t \in T$, we only require that $t_{c_1} = t_{c_2}$ if both c_1 and c_2 are inside of A or both c_1 and c_2 are outside of A; otherwise, t_{c_1} need not equal t_{c_2}.

Once the user has defined these four functions, our algorithms then look for the k regions that most strongly reject the null hypothesis. Logically, this is done by performing the computation given in Algorithm 1.

Example. Continuing with our antimicrobial resistance trend example, we would instantiate f, L, MLE_1, and MLE_0 as follows:

1. $f(A)$ would return the number of microbial infections in each cell in A for each year $(n_0, n_1, ...)$ as well as the number of resistant microbial infections for each cell in the area for each of the years $(k_0, k_1, ...)$.

2. For a cell c in a spatial area A, L accepts each n_y and k_y, as well as the rate of infection for the initial year p_0 and a change rate Δ. It then computes the binomial likelihood of the rates given the cell data:

$$L(\theta_c|X_c) \propto \prod_y [(p_0 + y\Delta)^{k_y} \times (1.0 - p_0 - y\Delta)^{n_y - k_y}]$$

3. MLE_0 performs a binomial MLE for an input spatial area. MLE_0 is run under the constraint that Δ is constant across all cells in the input area, and that $(p_0 + y\Delta) \in [0, 1]$. If the input area is the entire grid, this corresponds to the null hypothesis that the trend for each cell in the grid is the same, though the starting rate p_0 might be different.

4. Finally, MLE_1 performs two similar binomial MLEs, except that there are two different trends Δ_A and $\Delta_{\bar{A}}$ allowed for the areas within and without A, respectively.

3.4 So, What's the Difficulty?

Running the naive algorithm in Figure 1 can be quite expensive, but the computational difficulty is not necessarily associated with enumerating all of the local spatial regions; even for a 100×100 gird this should take only a few seconds. Rather, the computational difficulty is associated with *actually computing MLE_1 and MLE_0 for every candidate area in the grid*. Thus, we will consider pruning strategies that still enumerate all of the $O(n^4)$ local spatial areas in the grid, but can often inexpensively tell us before any MLEs are ever computed that it is impossible for the current LRT statistic to have a large or interesting value.

3.5 Remarks Regarding the Framework

First, we point out that this model is exceedingly general. The only constraint of any significance is that we require statistical independence of data generation across cells.

Second, there is the issue of statistical significance of the regions returned as the result of the search. Since our framework relies on the LRT, the null distribution is known once the likelihood function is defined and it is quite easy to associate a p value with each discovered region. The only difficulty is that an appropriate multiple-hypothesis-testing correction [4] must be used to account for the fact that $O(n^4)$ individual hypothesis tests have been performed.

If a user is uncomfortable with either the use of an asymptotic null distribution (because he or she is suspicious of relying on asymptotics) or the use of a multiple-hypothesis-testing correction (because he or she is worried that this will compromise the power of the underlying statistical test), then a Monte-Carlo method can be used to associate a p value with each result. Specifically, a large number of spatial grids can be generated under the null hypothesis, and the search for the most anomalous region can be performed in each. By counting how many of the grid replicas have a LRT statistic greater than the one obtained on the real data, one obtains an appropriate p value. The Monte Carlo method makes the pruning algorithms presented subsequently much more important. Since we are only interested in knowing how many of the grid replicas have larger LRT statistic than the one obtained on the real data, we will supply the largest LRT statistic from the real data as an initial cut off value to the remaining replicas, which may result in tremendous speedup comparing to the naive Monte Carlo method.

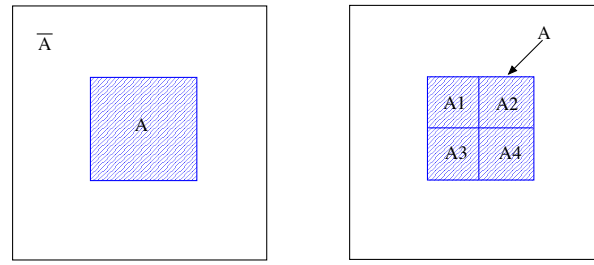

Figure 2: Illustration of bounding the maximum likelihood of region A by the product of the maximum likelihoods of subregion A_is. (a) The current testing region A. (b) The testing region A tiled by four subregions.

4. BASIC PRUNING MECHANISM

Given the framework from the previous section, the problem becomes speeding the search through each of the $O(n^4)$ possible regions in the grid. The basic idea that we will pursue is devising some way of immediately knowing—without performing any MLE—whether the LRT statistic can possibly exceed a given cutoff value for an area A.

So, how do we do this? At each iteration of Algorithm 1, we will consider a region A, and wish to know whether the value of Λ associated with A exceeds the cutoff. Normally, to compute Λ, we will invoke both MLE_1 and MLE_0. That is, we would compute $(\theta_A, \theta_{\bar{A}}) = MLE_1(A, f(G))$, and $\theta_G = MLE_0(f(G))$, and then compute the quantity $\Lambda = -2 \log L(\theta_G|X) + 2 \log L(\theta_A|X_A) + 2 \log L(\theta_{\bar{A}}|X_{\bar{A}})$

Our goal is to somehow avoid running the expensive MLEs and still obtain some idea about whether or not Λ exceeds the cutoff. Handling θ_G is easy. Since the value of $MLE_0(f(G))$ is the same no matter what the value of A, we can compute MLE_0 over the entire data set once, and then simply re-use this value for every iteration of the loop that enumerates all possible rectangles.

However, it is more difficult to avoid running $MLE_1(A, f(G))$. Fortunately, we can upper-bound the values of both $L(\theta_A|X_A)$ and $L(\theta_{\bar{A}}|X_{\bar{A}})$ obtained via MLE_1 by using the likelihoods associated with the MLEs that we have computed previously.

To do this, let $A = R_1 \cup R_2$ for non-overlapping R_1 and R_2, so that $X_A = X_{R_1} \cup X_{R_2}$. Let $\theta_A = \theta_{R_1} \cup \theta_{R_2}$ be the set of parameter values relevant to area A, that were computed via $MLE_1(A, f(G))$. Now, assume that θ'_{R_1} was computed directly by calling $MLE_0(f(R_1))$. Then we know that:

$$L(\theta_{R_1}|X_{R_1}) \leq L(\theta'_{R_1}|X_{R_1}) \quad (3)$$

This must be true since performing MLE_0 on a subregion of A has essentially relaxed the constraints for the optimization task performed by MLE_1 on A. Why? Recall that the shared parameters relevant to a given cell are categorized into two classes: the shared parameters that are not in the test set T, and the shared parameters that are in the test set T. θ_{R_1} is chosen by the optimization routine MLE_1 under two constraints:

1. All cells on the grid share the same set of values for all shared parameters not in the test set.

2. Consider the cells within A as a group, and cells within $\bar{A}$ as another group. Within both groups, all cells must share the same values for each test parameter, but across groups, test parameter values can differ.

Inequality 3 holds since θ'_{R_1} is chosen by MLE_0 under the relaxed constraints that:

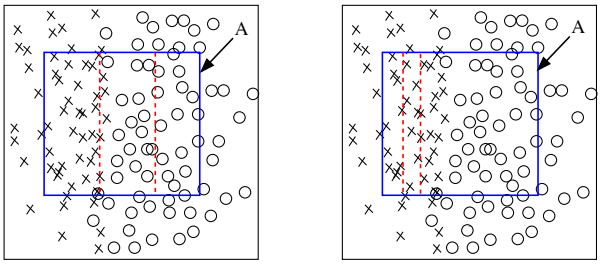

Figure 3: Illustration of the second tight bound criterion. (a) Three subregions evenly cover the target region. (b) Three subregions unevenly cover the target region, with one big subregion dominating the other two.

1. All cells within R_1 share the same set of values for all shared parameters not in the test set.

2. All cells within R_1 share the same set of values for all shared parameters in the test set.

Obviously, the former two constraints used by $MLE_1(A, f(G))$ to choose θ_{R_1} are more strict than the later two constraints used by $MLE_0(f(R_1))$ to choose θ'_{R_1}. As a result, inequality 3 must be true. Likewise, $L(\theta_{R_2}|X_{R_2}) \leq L(\theta'_{R_2}|X_{R_2})$. Thus:

$$L(\theta_A|X_A) \leq L(\theta'_{R_1}|X_{R_1}) \times L(\theta'_{R_2}|X_{R_2})$$

Thus, one can partition A into two arbitrary subregions, then invoke MLE_0 on each of those subregions independently, and use the result to upper-bound the value of $L(\theta_A|X_A)$ that would be obtained via a call to $MLE_1(A, f(G))$. Taking this to its logical conclusion, by induction, one can always give an upper bound to the value of $L(\theta_A|X_A)$ obtained via MLE_1 for $A = R_1 \cup R_2 \cup R_3 \cup ...$ by invoking $MLE_0(f(R_i))$ separately for each R_i (see an example in Figure 2). A similar argument holds for $L(\theta_{\bar{A}}|X_{\bar{A}})$.

Our basic tactic will be to pre-compute a number of strategically-chosen result sets from calls to MLE_0, and use those to attempt to avoid expensive calls to MLE_1 by upper-bounding the quantity of the result. If these upper bounds are tight, it will often be possible to prune A without ever calling MLE_1.

5. TIGHT BOUND CRITERIA

To bound $L(\theta_A|X_A)$ and $L(\theta_{\bar{A}}|X_{\bar{A}})$ we must be able to tile both A and $\bar{A}$ with a set of regions for which an MLE_0 has already been pre-computed. There are many possible precomputations and tilling methods, and the manner in which A (and $\bar{A}$) is tiled can have a significant effect upon the quality of the bound that is achieved.

In practice, there are two over-riding considerations when tiling a region in order to bound the value of $L(\theta_A|X_A)$:

1. The region should be tiled with as few tiles as possible.

2. For a fixed number of tiles, it is generally better to have a high variance in tile sizes than it is to have a low variance in tile sizes—that is, one big tile and $n-1$ small tiles are better than n medium-sized tiles.

It is easy to argue why adding more tiles than that are strictly needed is generally a bad idea. Under MLE_0, only one value for every parameter in the shared set is allowed. However, if A is covered with n tiles, then n different values of the parameters in the shared set are allowed, with one parameter value per tile. Thus, adding more and more tiles has the effect of adding more and more parameters. This tends to result in a looser and looser upper bound on $L(\theta_A|X_A)$.

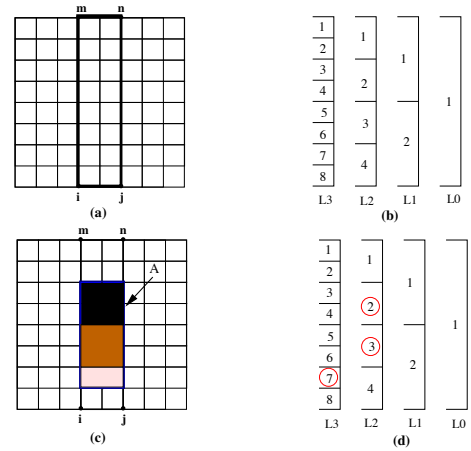

Figure 4: Illustration of tiling A. (a) Rectangle (m, n, i, j) is recursively split. (b) The splitting position in each level. (c) A is tiled using three precomputed rectangles. (d) The precomputed rectangles that are used (they are circled) to tile A.

The reason that large tiles are preferred—even if it results in one or two large regions and a few very small regions—is illustrated in Figure 3. The data points in this figure are represented by X's and O's; the data points represented by X's are generated by a different process than the O's and so they require different shared parameter values to model them than the O's do. If a number of smaller boxes are used (as in the left of Figure 3), then it is much more likely that the data inside of those boxes are homogeneous and contain mostly X's or mostly O's, and MLE_0 will produce a much looser bound than the case in the right of Figure 3.

6. PRECOMPUTATION AND BOUNDING

Were we to compute an MLE for every A in the grid, there would be $O(n^4)$ MLE computations. Our goal is to reduce this by at least a factor of n by performing only $O(n^3)$ MLE_0 pre-computations and a handful of actual $MLE_1(X, A)$ computations for those A's that are not pruned. Also, our goal is to limit those precomputation to smaller spatial regions, so they tend to be less expensive optimization problems. Since our reduced problem is to bound both $L(\theta_A|X_A)$ and $L(\theta_{\bar{A}}|X_{\bar{A}})$, we consider them separately.

6.1 Precomputation and Bounding for A

We devise a method that uses at most $2 \log \frac{n}{2}$ precomputed rectangles to tile any given A. The precomputed rectangles are obtained as follows. Consider the *biggest* rectangle that is enclosed by a pair of the vertical grid lines. Figure 4(a) illustrates one such rectangle (m, n, j, i). The horizontal grid line which crosses the center point of the grid is used to split this rectangle into two identical subrectangles. If we recursively split the resulting subrectangles by horizontal grid lines from their midpoints until we reach the lowest resolution of the grid, our precomputation set contains $\frac{(2n-1)(n+1)n}{2}$ rectangles, which is in the order of $O(n^3)$.

To use the precomputed rectangles to tile any given A, we use a divide and conquer method. Figure 5 gives the detailed algorithm. The recursive function **Tile_A** accepts the bottom-left (x_1, y_1) and top-right (x_2, y_2) coordinates of a rectangle A, and a pair of low and high values (initially, low is set to zero and high is set to n). It splits A in a top-down fashion following the split points used during precomputation. This method will always tile A with the biggest possible rectangle in our precomputed set and guarantees

Procedure Tile_A(x_1, y_1, x_2, y_2, low, high)
1. **Let** mid=(low+high)/2
  //base case
2. **If** ((y_1 ==low && y_2 ==high) **or** (y_1 ==low && y_2 ==mid) **or** (y_1 ==mid && y_2 ==high))
  3. **Return** rect(x_1, y_1, x_2, y_2) /* rect() returns the stored rectangle log-likelihood */
  //recursive case
4. **If** ($y_2 \leq$mid)
  5. **Return Tile_A**(x_1, y_1, x_2, y_2, low, mid)
6. **Else If** ($y_1 <$mid **and** $y_2 >$mid)
  7. **Return Tile_A**(x_1, y_1, x_2, mid, low, mid)+ **Tile_A**(x_1, mid, x_2, y_2, mid, high)
8. **Else If** ($y_1 \geq$mid)
  9. **Return Tile_A**(x_1, y_1, x_2, y_2, mid, high)

Figure 5: The algorithm for tiling A

we use the smallest number of tiles. Figure 4 (c) and (d) illustrate an example where we use two level 2 rectangles (numbered 2 and 3) and one level three rectangle (numbered 7) to tile the given A.

6.2 Precomputation and Bounding for $\bar{A}$

Now we turn our attention to the tiling methods for $\bar{A}$. There are two different tiling strategies that we employ:

The radial method. As illustrated in Figure 6 (a), in a clock-wise order, we elongate the edges of a rectangle A until the edges hit the grid borders. $\bar{A}$ is then divided into four rectangles denoted $\bar{A}_1$ to $\bar{A}_4$, which are used to tile $\bar{A}$. We can do the same thing in a counter-clockwise order, which is depicted in Figure 6 (b). In order to tile any given $\bar{A}$ using the radial method, the precomputed set should contain all the rectangles that share at least one corner with the grid. This set can be obtained by considering all of the intersection points on the grid. We connect each intersection point on the grid with the four corners of the grid. This produces four diagonals, each of which creates one rectangle in our precomputed set. Since there are $O(n^2)$ intersection points, there are $O(n^2)$ rectangles in our precomputed set.

The sandwich method. As illustrated in Figure 6 (c), if we elongate the two vertical edges of A in both directions until they reach the borders of the grid, $\bar{A}$ is divided into four rectangles, denoted $\bar{A}_1$ to $\bar{A}_4$. We use these four resulting rectangles to tile $\bar{A}$. In the same fashion, we can do this horizontally as illustrated in Figure 6 (d). In order to tile any $\bar{A}$ using this method, we need to precompute all the rectangles that share two corners with the grid. In Figure 6 (c), these rectangles are $\bar{A}_2$ and $\bar{A}_4$. Notice that these two rectangles are in the the precomputed set for the radial method, so we can reuse them. Also, since we want to avoid any additional precomputations to bound $\bar{A}_1$, we call the **Tile_A** procedure from the previous subsection to upper bound $\bar{A}_1$. We can upper bound $\bar{A}_3$ in similar fashion. As a result, with no additional precomputation, we can obtain the bounds using the sandwich method.

We have discussed four methods to bound $\bar{A}$ (Fig 6); in practice, we compute each and choose the tightest bound on $L(\theta_{\bar{A}}|X_{\bar{A}})$.

7. FINAL SEARCH ALGORITHM

The final search algorithm in Figure 7 modifies the naive algorithm. At each iteration, the new algorithm obtains an upper bound for the current LRT statistic, and compares the upper bound with the current cutoff related to the k^{th} largest Λ discovered so far. If the upper bound is less than the cutoff, the current region is pruned. Otherwise, the exact value for the LRT statistic is computed.

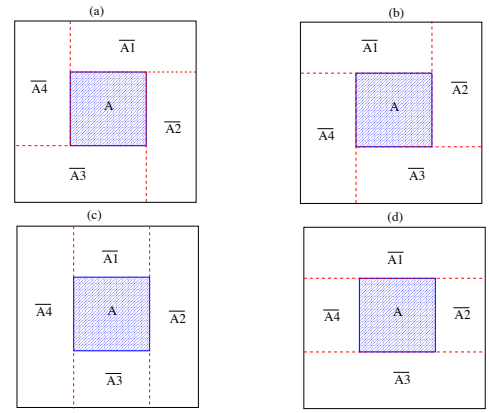

Figure 6: Tiling methods for $\bar{A}$. (a) and (b) depict the Radial methods. (c) and (d) depict the Sandwich methods.

Input: a spatial grid G, f, MLE_0,MLE_1, L and k
Output: top-k anomalous regions.
1. Precompute the $O(n^3)$ rectangles described in Section 6.1
2. Precompute the $O(4n^2)$ rectangles described in Section 6.2
3. **Let** $\theta_0 = MLE_0(f(G))$.
4. **For** each rectangle A on the grid:
  5. Follow Section 6.1 to get the upper bound for $\log L(\theta_A|X_A)$
  6. Follow Section 6.2 to get the upper bound for $\log L(\theta_{\bar{A}}|X_{\bar{A}})$
  7. Combine the results of step 3, 5, 6 to an upper bound for Λ_A
  8. **If** the upper bound is less than the k^{th} best, prune A
  9. **Else** compute Λ_A; if in the top k, remember A

Figure 7: Top-k LRT statistic search with pruning

8. EXPERIMENTS

8.1 Experiment One: Synthetic Data

Experimental Goal. Our goal is to answer two questions:

1. Does our precomputation actually cut down on the number of MLEs that need to be run in a realistic setting?

2. Second, the LRT framework has a known asymptotic null distribution. If we use this fact along with a standard Bonferroni correction to take into account the multiple hypothesis test (MHT) problem induced by running a separate hypothesis test for each area in an $n \times n$ grid, will we (a) still be able to detect any subtle anomalies, and (b) be able to correctly recognize those cases where there is no anomaly in the underlying data?

Experimental Setup. For our experiments over synthetic data, we used a simple binomial likelihood model. This model is chosen because performing the required MLE is fast, so we can repeatedly run and re-run tests over reasonably large grids.

Our setup is as follows. For each cell c in the grid, we randomly generate a population size n_c. Then the number of "successes" k_c in the cell is generated by sampling from a Bin(n_c, p) random variable for success rate p. Given a testing rectangle A, denote the "success" rate within A by p and within $\bar{A}$ by q. The test set is p, the local parameter is the number of binomial trials in a particular cell. The null hypothesis asserts that $p = q$, and the alternative hypothesis asserts that $p \neq q$.

Since the complete parameter space only has one more dimension than the null parameter space, the null distribution is chi-

Table 1: Average pruning rates and accuracy for 50 trials on a 128×128 grid. The pruning rate is $\frac{\text{pruned rectangle \#}}{\text{total rectangle \#}}$.

Test	Avg Pruning Rate	Accuracy
Null (standard)	99.9994%	no false alarm
Null(city population)	99.9996%	no false alarm
Hot spot $p = 0.003$	99.9712 %	100%
Hot spot $p = 0.01$	99.9722 %	100%

Table 2: Average pruning rates and accuracy for 50 random trials on a 256×256 grid.

Test	Avg Pruning Rate	Accuracy
Null (standard)	99.9999%	no false alarm
Hot spot $p = 0.003$	99.9996%	100%

squared with one degree of freedom. For our tests, we run 50 trials on a grid size of 128×128. To test the scalability, we also consider a grid size of 256×256. Our initial cutoff value for pruning was chosen to correspond to an overall false positive rate of 5%. In all tests, we seek the top subregion.

We ran two different sets of experiments where the null hypothesis was in fact true. In the first, the underlying n_c values are relatively uniform, where n_c was always sampled from a Normal($\mu = 1 \times 10^4$, $\sigma = 1 \times 10^3$) distribution. The "success" rate for each cell was set to 0.001. In the second case, there was a single densely-populated region, or simulated "city". We randomly selected an area of size 12×12, and within this area, n_c was always sampled from a Normal($\mu = 1 \times 10^5$,$\sigma = 5 \times 10^3$) distribution. We recorded the pruning rate (the number of rectangles pruned/the total rectangles), and checked if there were false alarms.

We also simulated a case where the alternative hypothesis holds. The setup was the same as the standard null case, except that we randomly pick a 4×3 region as the "hot spot". In one set of tests (the "subtle anomaly case"), the success rate for generating each k_c was set to 0.003. In the other (the "extreme anomaly case"), the success rate was set to 0.01. Results are given in Tables 1 and 2.

Discussion. In general, the results shows very high pruning power. Taking into account the precomputated MLEs, the pruning rates realized on the 128×128 grid would on average reduce the number of tests required by a factor of 31.1 to 31.4 compared to the naive algorithm. On the 256×256 grid, our framework would reduce the number of tests required by a factor of 63.4.

Also, with an overall 0.05 level test, our framework did not find any false alarms in tests where the null hypothesis held. The total lack of false alarms (even at a significance level of 0.05) may result from our application of a conservative Bonferroni correction. On the other hand, in both the "subtle anomaly" and "extreme anomaly" cases, the framework had 100% detection accuracy. Overall, these results show that the framework seems to be both safe and effective.

8.2 Experiment Two: The ARM Database

Application Description. 357 U.S. hospitals participate in the ARM program, where each year, for each (bacteria/antimicrobial drug) combination, each hospital submits the number of isolates (bacteria instances) tested, as well as the number of times that the bacteria was found to be susceptible to the given drug. As described in the introduction, our goal is to find local spatial regions where the *temporal trend* of antimicrobial resistance change within the region is significantly different than the trend outside of the region.

Experimental Goal. We wish to experimentally test whether our LRT framework can be used to effectively speed the naive algorithm. There are two primary questions we wish to answer:

1. First, what is the pruning rate of our LRT framework over the real data set, with a realistic likelihood model?

2. Second, what is the speedup in wall-clock running time compared to the naive algorithm?

Instantiating the LRT Framework. To make use of LRT framework, and following the notations introduced in the running example of Section 3.3, we provide instances of f, L, MLE_0 and MLE_1 as follows:

1. For each cell on the grid, there may be one or more hospitals. Function f reports the number of aggregate microbial infections in each cell in a given area A for years 0, 1, ... (n_0,n_1,...) as well as the number of resistant infections in each year(k_0,k_1,...) for all the hospitals in the cell over a fixed five years span. That is, if x indexes the cell, and y indexes the year, then f reports (n_{xy}, k_{xy}) for every x and y.

2. For each cell on the grid, we treat each year's data as a single binomial trial. L accepts the cell's summary statistics reported by f as well as the rate of infection for the initial year p_0 and a change rate Δ :

$$L(\theta_c|X_c) = \prod_y \left(\begin{array}{c} n_y \\ k_y \end{array} \right) (p_0+y\Delta)^{k_y}(1.0-p_0-y\Delta)^{n_y-k_y}$$

Here, we assume there is a linear relationship for each cell's yearly infection rates. That is, $p_y = y\Delta + p_0$, and $p_y \in [0, 1]$. Since we are testing if any subset of adjacent cells exhibits a different trend than the rest of the cells, the test set is Δ, and the local parameter is p_0 and n_y.

Plugging in the cell's PDF into Equation 2, we have the entire grid's likelihood:

$$\begin{aligned} L^*(\theta|f(G)) \quad = \quad & \prod_{x\in A}\prod_y [\left(\begin{array}{c} n_{xy} \\ k_{xy} \end{array} \right) ((p_0)_x + y\Delta_A)^{k_{xy}} \times \\ & (1.0 - (p_0)_x - y\Delta_A)^{n_{xy}-k_{xy}}] \times \\ & \prod_{x\in \bar{A}}\prod_y [\left(\begin{array}{c} n_{xy} \\ k_{xy} \end{array} \right) ((p_0)_x + y\Delta_{\bar{A}})^{k_{xy}} \times \\ & (1.0 - (p_0)_x - y\Delta_{\bar{A}})^{n_{xy}-k_{xy}}] \end{aligned} \quad (4)$$

In Equation 4, $\theta = \{\Delta, n_{xy}, (p_0)_x\}$. Assuming that all the cells within a test region A share the same trend Δ_A, and all the cells outside region A share the same trend $\Delta_{\bar{A}}$, then we have the following competing hypotheses:

- H_0: $\Delta_A = \Delta_{\bar{A}}$
- $H_a : \Delta_A \neq \Delta_{\bar{A}}$

3. MLE_0 implements a numerical optimization routine for maximizing the value of L in Equation 4. Numerical methods are required due to the lack of an analytic solution.

4. MLE_1 simply invokes MLE_0 on $f(A)$ and $f(\bar{A})$ independently, since each shared parameter is also in the test set.

Table 3: LRT framework time (in days) vs. naive algorithm for "trend" model.

$n \times n$	LRT time	LRT pruning	Naive time	Speedup
16×16	0.15	96.5286%	2.6	17.3
32×32	1.13	97.6303%	35.9	31.8
64×64	11.9	98.0431%	544	45.7

Table 4: LRT framework time (in days) vs. naive algorithm for "wage" model.

$n \times n$	LRT time	LRT pruning	Naive time	Speedup
16×16	0.29	78.8396%	1.28	4.41
32×32	2.01	86.1219%	14.0	6.97

Experiment Setup. For the bacteria-antimicrobial combination of S. aureus and Nafcillin, we extracted data from the ARM database for 203 hospitals that provided data in the time period from 2000 to 2004, inclusive. All of the hospitals were organized into an $n \times n$ spatial grid. We first sort the hospitals using the longitude of their physical locations. They are then grouped into n equi-depth buckets. The placement of hospitals into the n buckets determines which column of the grid each hospital belongs to. Similarly, we sort all hospitals on latitude, partition them n ways, and use the partitioning to determine the rows.

We tested three different grid sizes: $n = 16, 32$, and 64, and use our framework to find the spatial region that most strongly rejects the null hypothesis. For each grid size, we recorded both the pruning rate and the wall-clock running time. Since MLE_0 requires one second over the entire data set, there is a strong relationship between the pruning rate and running time; enumerating all of the cells in the grid is inexpensive compared to running one or more MLEs for each area in the grid. Also, due to the high cost of running the MLE, in order to compare our methods with the naive method of simply running an MLE over each area, we had to resort to sampling. For each iteration of the naive method of Algorithm 1, before we invoke MLE_0 over A and $\bar{A}$, we flip a coin having a 1% chance of obtaining a "head" result. If the result is a "head", we compute the LRT statistic by running the MLE. Otherwise, we skip the rectangle. While the result of running this sampling-based algorithm will be useless, the running time is expectedly $\frac{1}{100}$ of the time required to run the naive algorithm to completion. All results are presented in Table 3.

Discussion. We observed a speedup of between one and two orders of magnitude with respect to the naive algorithm, with the speedup increasing with increasing grid size. The results are quite striking. For example, in the 64×64 case, our framework took about 12 days to finish running. On the other hand, the naive method is estimated to take 544 days, or about one and a half years! Thus, in this real application, the pruning strategies that the proposed framework utilizes can turn a computation that is totally infeasible to one that may still be expensive—but is possible in a time frame that is likely acceptable in most epidemiological settings.

8.3 Experiment Three: Census Data

Application. Our data comes from a 5% sample of the 2000 U.S. census data. [2] Each database record contains data describing a single person, where we know (a) the person's annual wage, (b) the person's highest educational attainment level, and (c) the place where the person works. Annual wage is an integral number. There are eighteen classes of educational attainment, such as Bachelor's degree, Master's degree, etc. For privacy reasons, the place of work attribute in this data is described in terms of the Public Use Microdata Area (PUMA), a Census Bureau-defined area of 100,000+ residents; we use the center of each PUMA to place each person into a spatial grid placed over the U.S.A. Given this data, we wish to ask the following question:

Are fluctuations in the level of income across the country totally explained by differences in education levels across the country, or are there real differences in income level according to the spatial region where people work?

This question can be re-cast in a slightly different way that is quite amenable to analysis using our proposed framework:

If we condition each person's income level upon the level of education that they obtained, are there spatial subregions of the country where the income level is significantly different from the rest?

Experimental Goal. We wish to experimentally test whether our LRT framework can be used to effectively speed a realistic, large-scale data analysis problem—there are 1.5 million records in the set we consider. Furthermore, the problem is made even more realistic by the fact that the data are not clean—some of the data have a missing label for the level of educational attainment, which is an issue that must be dealt with in a statistically rigorous fashion; we use an EM algorithm to deal with the missing data.

Instantiating the LRT Framework. To use the LRT framework, again we need to provide the following components:

1. For each cell on the grid, there may be co-located many PUMAs. Function f reports each respondent's annual wage, together with the person's education level whose values is in $\{0, 1, \ldots 18\}$, where 0 indicates the respondent's education level is missing. For a cell in area A, denote the set of labeled data returned by f by X_L, and denote the set of data with missing education level (the unlabeled data) using X_U. Each element of $X_L \cup X_U$ takes the form (x, y) where x is the reported income and y indicates educational level.

2. For each cell, we describe the wages distribution using a Gamma mixture model, where there are 18 component distributions, each of which represents the wage distribution of an educational class. If a data point has a valid education label, we use the Gamma component distribution to get the data point's likelihood directly. Otherwise, we plug in the data point into the mixture model. We use w_j to denote the weight of the j^{th} component for a cell; this is the probability that an arbitrary person in the current cell achieves an education level of j. (α_j, β_j) denotes the Gamma distribution's parameters for the j^{th} component. Then, L is:

$$L(\theta_c|X_c) = \prod_{(x,y)\in X_L} p(x|\alpha_y, \beta_y) \times \prod_{x\in X_U} \sum_{j=1}^{18} w_j p(x|\alpha_j, \beta_j)$$

   Here, we are interested in testing if the *component distribution* representing each education class is the same across the country. The test set includes (α_j, β_j) for all possible j, the local parameters are the weights w_j's for each mixture component j in a cell.

   Again, we can plug in the cell's PDF into Equation 2 to get the grid's likelihood function $L^*(\theta|f(G))$. Using i to index the cells, we have $\theta = \{(\alpha_j^A, \beta_j^A), (\alpha_j^{\bar{A}}, \beta_j^{\bar{A}}), w_{ij}\}$ for every i and j. Here, the test set is (α_j, β_j), the local parameter is w_{ij}.

[2] http://usa.ipums.org/usa/

Assuming all the cells within a testing region A share the same education class distribution Gamma$(x|\alpha_j^A, \beta_j^A)$, and all the cells without A share the same class distribution Gamma$(x|\alpha_j^{\bar{A}}, \beta_j^{\bar{A}})$ for each $j \in \{1, \ldots, 18\}$. Then, we have the following competing hypothesis:

- H_0 :$\alpha_j^A = \alpha_j^{\bar{A}}$ and $\beta_j^A = \beta_j^{\bar{A}}$ for all $j \in \{1, \ldots, 18\}$.
- H_a: $\exists\, \alpha_j^A \neq \alpha_j^{\bar{A}}$ or $\beta_j^A \neq \beta_j^{\bar{A}}$ for some $j \in \{1, \ldots, 18\}$.

3. MLE_0 in this problem is an implementation of the EM algorithm [3], which is chosen so that we can handle the MLE over the unlabeled data in a statistically meaningful fashion, without simply throwing them away. Since EM only converges to a local maxima, our implementation performs several random restarts, though in practice we find that no matter what the starting point, the end quality of MLE_0 does not change significantly enough to affect the LRT statistic. For brevity, we omit the details of the derived EM.

4. MLE_1 is two invocations of MLE_0 with X_A and $X_{\bar{A}}$, respectively.

Experimental Setup. We extracted data for fourteen, contiguous U.S. states, forming a rectangle from Arizona to Indiana, covering 330 PUMAs. The 330 PUMAs were organized into an $n \times n$ grid using a method identical to the one used for the ARM data. We used $n = 16$ and 32. We used our framework to look for the top subregion—that is, the one with the largest LRT statistic value. Since MLE_0 relies on an iterative EM implementation, it is expensive to run. We used sampling to obtain the estimated wall-clock time for the naive method. The results are presented in Table 4.

Discussion. We find that we still achieve good performance compared to the naive method, even in a small grid size. However, the pruning rate and speedup, while significant, are not as dramatic as with the other two data sets. It appears that there is some particular aspect of this application which results in the bound computed by the framework not being quite as tight as in the other two cases. Still, in the case of the 32×32 grid, the framework is able to reduce the required time from two weeks down to two days.

9. RELATED WORK

Detecting anomalous spatial regions have been a popular research topic. Two lines of research have been popular in this area. One is spatial clustering, which includes CLARANS [10], DBSCAN [5], and STING [11] etc. The other group focuses on the performance study on detecting statistical significant spatial or spatial-temporal cluster (hot spot) [8, 9, 2, 1]; most papers in this second line of work are based on Kulldorff's spatial scan statistic (SSS)[6, 7]. The SSS is used to detect non-random spatial clusters of event occurrence. Conditioned on the observed event counts, and assuming event independence, the SSS is expressed in the following form:

$$C_A \log \frac{C_A}{P_A} + (C_G - C_A) \log \frac{C_G - C_A}{P_G - P_A} - C_G \log \frac{C_G}{P_G}$$

where C_A is the aggregate event count for area A, and C_G is the aggregate event count for the entire spatial region. P_A and P_G are the population sizes for zone A and the entire spatial region, respectively. Monte Carlo methods is employed to obtain the empirical null distribution of the test statistic.

Since obtaining null distribution is expensive for Kulldorff's spatial scan statistic, several methods have been proposed for addressing the performance issue [9, 8, 1, 2]. These methods differ from our own in that they aim to actually avoid considering all $O(n^4)$ rectangular areas, whereas our goal is to avoid expensive LRT statistic computations. The existing methods in the literature only apply to those relatively simple density measures that are convex or monotonic with respect to the ratio of zone population over entire population and the ratio of zone's event count over the entire event count. For such measures, the scan-statistic-based approach would be preferred to our own. Unfortunately, existing methods do not apply to the likelihood functions used in the trend or wage model considered in this paper; in these cases, our algorithms are the only option.

10. CONCLUSIONS

In this paper, we have considered the problem of detecting spatial anomalies in an $n \times n$ grid using a LRT with a chi-squared asymptotic null distribution. Our focus is on using pruning to reduce the magnitude of the constant c in the underlying $O(cn^4)$ computation in the case where c is large due to an expensive LRT statistic. Experiments in real-life applications show that our methods are effective at reducing c by almost two orders of magnitude.

Acknowledgements. Material in this paper was supported by the National Science Foundation under grant no. 0612170.

11. REFERENCES

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

[2] D. Agarwal, J. M. Phillips, and S. Venkatasubramanian. The hunting of the bump: On maximizing statistical discrepancy. In *SODA 2006*, pages 1137–, 2006.

[3] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal Royal Stat. Soc., Series B*, 39(1):1–38, 1977.

[4] S. Dudoit, J. P. Shaffer, and J. C. Boldrick. Multiple hypothesis testing in microarray experiments. *Statistical Science*, 18:71–103, 2003.

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

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

[7] M. Kulldorff. Spatial scan statistics: model, calculations, and applications. In *Scan Statistics and Applications*, pages 303–322, 1999.

[8] D. B. Neill and A. W. Moore. A fast multi-resolution method for detection of significant spatial disease clusters. In *NIPS 2003*, pages 256–265, 2003.

[9] D. B. Neill and A. W. Moore. Rapid detection of significant spatial clusters. In *SIGKDD 2004*, pages 256–265, 2004.

[10] R. Ng and J. Han. Clarans: A method for clustering objects for spatial data mining. *IEEE Trans. Knowl. Data Eng.*, 14(5):1003–1016, 2002.

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

[12] S. S. Wilks. The large sample distribution of the likelihood ratio for testing composite hypotheses. *Annals of Mathematical Statistics*, 9:60–62, 1938.