
3 A Sensor Network Data Model for the Discovery of Spatio-Temporal Patterns*

*Betsy George, James M. Kang,
and Shashi Shekhar*

Department of Computer Science and Engineering
University of Minnesota

CONTENTS

Introduction	16
Application Domain	18
Related Work	20
Basic Concepts	21
Spatio-Temporal Sensor Graph	21
Case Studies	23
Anomaly Detection	23
Definition	23
Application	24
Method	24
Execution Trace	25
Computational Complexity	26
Basic Hotspot Detection	26
Definition	26
Application	26
Method	26
Execution Trace	28
Computational Complexity	28
Growing Hotspot Detection	28
Definition	28
Application	29
Method	29
Execution Trace	29
Computational Complexity	31

*This work was supported by an NSF-SEI grant, NSF-IGERT grant, Oak Ridge National Laboratory grant, and US Army Corps of Engineers (Topographic Engineering Center) grant. The content does not necessarily reflect the position or policy of the government and no official endorsement should be inferred.

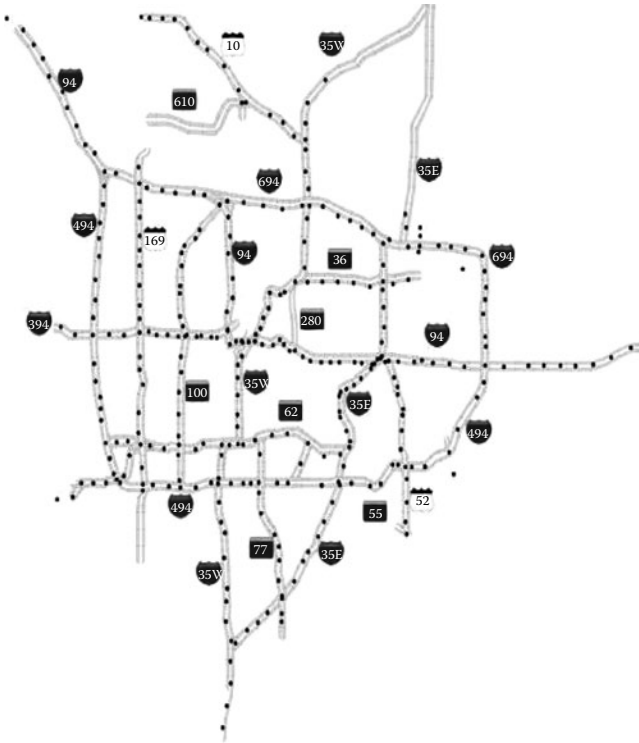
Conclusion and Future Work	31
Acknowledgments	32
Biographies	32
References	33

ABSTRACT

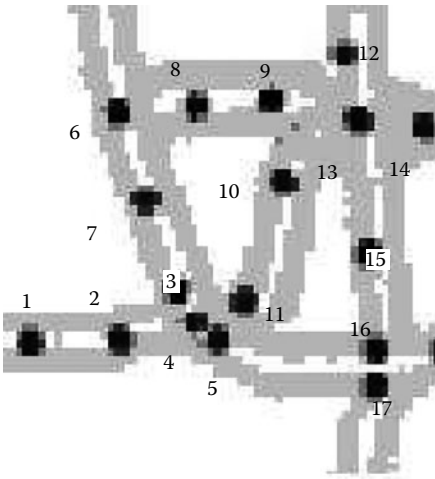
Developing a model that facilitates the representation and knowledge discovery on sensor data presents many challenges. With sensors reporting data at a very high frequency, resulting in large volumes of data, there is a need for a model that is memory efficient. Sensor networks have spatial characteristics which include the location of the sensors. In addition, sensor data incorporates temporal nature, and hence the model must also support the time dependence of the data. Balancing the conflicting requirements of simplicity, expressiveness, and storage efficiency is challenging. The model should also provide adequate support for the formulation of efficient algorithms for knowledge discovery. Though spatio-temporal data can be modeled using time expanded graphs, this model replicates the entire graph across time instants, resulting in high storage overhead and computationally expensive algorithms. In this chapter, we discuss a data model called Spatio-Temporal Sensor Graphs (STSG) to model sensor data, which allows the properties of edges and nodes to be modeled as a time series of measurement data. Data at each instant would consist of the measured value and the expected error. Also, we present several case studies illustrating how the proposed STSG model facilitates methods to find interesting patterns (e.g., growing hotspots) in sensor data.

INTRODUCTION

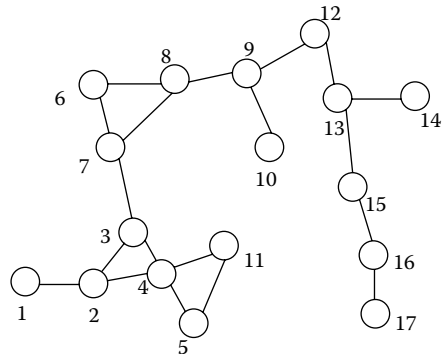
Finding novel and interesting spatio-temporal patterns in the ever increasing collection of sensor data is an important problem in several scientific domains. Many of these scientific domains collect sensor data in outdoor environments with underlying physical interactions. For example, in environmental science, a timely response to anticipated watershed/in-plant events (e.g., chemical spill, terrorism) to maintain water quality is required. Such a case occurred in Milwaukee, WI, in 1993 where a harmful pathogen (called *Cryptosporidium parvum*) outbreak occurred in the river streams that infected more than 400,000 people with more than 100 deaths. The estimated total cost for the treatment of outbreak-related illness was \$96.2 million [7]. As was the case in Milwaukee, such failures typically are detected long after the exposure by observed spikes in doctor/hospital visits or sales of certain medicines. In addition to unplanned “natural” events like the *Cryptosporidium* episode, another concern regarding water supplies is an act of terrorism. Clearly, when public health is at stake, waiting for the illnesses and fatalities to arise is much too late and identifying and modeling these spatio-temporal patterns such as hotspots and growing hotspots from sensor graphs is important [14]. Other applications that generate similar sensor data may be traffic road systems where measurements of traffic flow and congestion are important, especially in emergency operations such as evacuations.



(a) Sensors on Twin Cities road network



(b) Neighborhood of US169, TH62, 1494



(c) Sensor graph for sensors in Figure 3.1(b)

FIGURE 3.1 Sensor networks periodically report time-variant traffic volumes on Twin Cities, MN highways. (Best viewed in color, Source: Mn/DOT)

A collection of sensors may be represented as a sensor graph where the nodes represent the sensors and the edges represent selected relationships. For example, sensors upstream and downstream in a river may have physical interactions via water flow and related phenomenon such as plume propagation. Relationships can also be geographical in nature, such as proximity between the sensor units. As an example, [Figure 3.1\(a\)](#) shows a layout of traffic sensors in the Twin Cities, MN. The graph representation of a part of this layout [given in [Figure 3.1\(b\)](#)] is shown in [Figure 3.1\(c\)](#). The nodes of the graph represent the sensors and the edges represent the physical relationships between the various sensors. In this example, the edges are based on the proximity between the sensors.

Formulation of a model to represent a sensor graph that supports mining useful information from data poses some significant challenges. Since the volume of data is large, the model used to represent the sensor graph must be storage efficient. It should also provide sufficient support for the design of correct and efficient algorithms for data analysis. Second, the sensor graph characteristics modeled as pairs, <measured value, error>, can be time-dependent (e.g., the flow rate in a river stream). The model used to represent a time-dependent graph should be able to represent the time-variance, simultaneously maintaining the storage efficiency.

A sensor graph is spatio-temporal in nature since the relative locations of the sensor nodes and the time-dependence of their characteristics are significant. Spatio-temporal graphs can be modeled as time-expanded graphs, where the entire network is replicated for every time instant [17]. The changes in the graph can be very frequent and for modeling such frequent changes, the time expanded networks would require a large number of copies of the original network, thus leading to network sizes that are too memory expensive. Moreover, while modeling sensor graphs that involve no physical flow, a direct application of this model might not be possible.

A Spatio-Temporal Sensor Graph (STSG) models the changes in a spatio-temporal graph by collecting the node and edge attributes into a set of time series. The model can also account for the changes in the topology of the network. The edges and nodes can disappear from the network during certain instants of time and new nodes and edges can be added. A Spatio-Temporal Sensor Graph keeps track of these changes through a time series attached to each node and edge that indicates their presence at various instants of time. The stochastic nature of the physical relationships between the sensors (e.g., the flow rate of the river stream that connects the sensors) is accounted for by expressing each element in the attribute time series as a pair of values (i.e., <measured value, error>). Several case studies are also provided to validate the model in the context of discovery of spatio-temporal patterns from sensor data. Analysis shows that this model is less memory expensive and leads to algorithms that are computationally more efficient than those for the time-expanded networks.

APPLICATION DOMAIN

Modeling spatio-temporal graphs has significant applications in a number of scientific application domains. Discovering knowledge from the large amount of data collected from sensors can be used in predicting trends in domains such as environmental science, thus emphasizing the need for a model. Transportation network flow

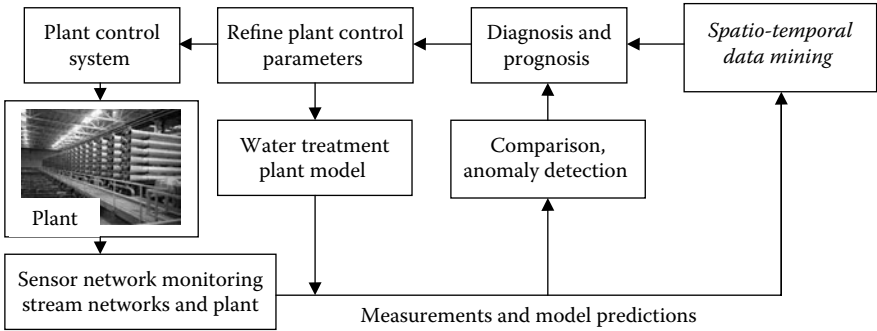


FIGURE 3.2 Spatio-temporal data mining in transformative water quality.

patterns are being increasingly monitored by sensors. The data can be used to find routes that are frequently congested and can be used in network planning. Since varying levels of congestion can lead to time-dependent travel times on road segments, the road network represented by a spatio-temporal graph might give more accurate results for routing queries such as shortest paths. Accounting for time dependence in transportation networks would make evacuation-planning algorithms in emergency planning generate results that are more accurate.

The role of spatio-temporal data mining in the environmental sciences is shown in Figure 3.2. The figure gives an example of the water flow starting from a water treatment plant to a sensor network collecting data throughout the watershed. The data collected from the sensor network is handled in two parts: (a) the collected data are analyzed for any interesting patterns (e.g., anomalies) and if any are found, a diagnosis and prognosis are made; and (b) the collected data are used for learning of any new and interesting spatio-temporal patterns (e.g., growing hotspots) resulting in the refinement of the diagnosis and prognosis rule base. Based on the prognosis, the treatment plant can readjust any necessary parameters for the plant control system and plant models. Such a spatio-temporal data mining system could be used to issue warnings to recreational users of water resources (e.g., beach closings) and to optimize water treatment conditions to protect the public and sensitive water treatment infrastructure. For example, similar to weather forecasts, a water quality prediction for beaches could be provided (e.g., 30% likelihood that coliform levels will be exceeded).

Similarly, if a spike in pathogen or hazardous chemical concentration is predicted, water intake could be suspended temporarily, processes could be adjusted in real time, or an additional treatment process could be brought online. The discovery of interesting patterns within sensor data for outdoor application domains is often arduous and complex. Many challenges [6,8,18,21] and hurdles need to be overcome. One challenge is to support remote monitoring of sensor networks distributed over an area, to check the overall functioning of the system as well as to detect interesting events related to measurements. Examples include the tasks of identifying malfunctioning sensors or interesting events.

In general, there are two types of outdoor sensor networks [1,6,8,11,13,18,21]. The first type is a *wired* sensor network, for example, traffic management center at

the Minnesota Department of Transportation [25,26] that consists of the following: (a) sensors that are wired within the Twin Cities, MN highway system and (b) a very large network containing over 4000 sensors across a 20-mile radius in the Twin Cities and sampled every 5 minutes, and [1] a real-life application in a metropolitan area. The second type is a *wireless* sensor network, for example, the one used by the Water Resources community that has been recently deployed at the Minnehaha Creek in Minnesota that consists of the following: (a) a set of sensors placed in the environment communicating wirelessly among each other, (b) an initial deployment of a handful of sensors and a future goal of increasing the number of sensors to monitor the Mississippi River, and (c) a live application in the natural environment [14].

RELATED WORK

Models have been proposed for sensor data [2,4] where the characteristics of the sensor devices are classified as stored data and the data collected is represented as are time series. These models do not fully represent the connectivity among the sensors. Graphs have been used to represent collection of sensors and to formulate algorithms for applications such as routing and location tracking [12,19]. Recent research in spatial anomaly detection proposed in [25,26] used sensor datasets structured as graphs. An accurate representation of sensors should include the spatial attributes and time-dependent parameters of the graph. Most graph representations ignore the time-dependence of the attributes. Some knowledge discovery algorithms are limited due to the ignoring of spatial relationships [3,15]. Some work has focused on managing the datasets produced by *wired* and *wireless* sensor networks using “spatial time series” [27].

Traditionally, spatio-temporal networks such as transportation networks have been modeled using time-expanded graphs [16,17,20]. This method duplicates the original network for each discrete time unit $t = [0, 1, \dots, T]$, where T represents the extent of the time horizon. The expanded network has edges connecting a node and its copy at the next instant in addition to the edges in the original network, replicated for every time instant. This approach significantly increases the network size and is very expensive with respect to memory. Because of the increased problem size due to replication of the network, the computations become expensive. In addition, time-expanded graphs cannot model sensor networks in cases where there is no flow parameter involved. In such cases, the cross edges that represent a flow from one node to another lose significance.

A model called Time Aggregated Graph (TAG) was proposed to represent spatio-temporal networks [9,10]. This model aggregates time-dependent parameters on edges and nodes to time-series attributes rather than replicate the entire graph for each time instant. TAG can also model topological changes that could occur in the graph (e.g., disappearance of an edge during a time interval) over time. This model has been used in formulating routing algorithms for transportation networks with time-dependent travel times, such as shortest path computation and best start time computation [9]. Analysis shows that the model is more memory efficient than time-expanded graphs. According to the analysis in [23], the memory requirement for a time-expanded network is $O(nT) + O(n + m)T$, where n is the number of nodes

and m is the number of edges in the original graph. The memory requirement for the time-aggregated graphs would be $O(m + n)T$. Since the physical relationships among sensors are stochastic in nature, there is a need to model the probabilistic characteristics of edges and nodes, which is not modeled in TAG. Spatio-Temporal Sensor Graph represents stochastic parameters by specifying the measurement and the expected error. Each attribute value would be a pair, <measured value, error>. The STSG model can be used as the basis for algorithms for hotspot discovery and growing hotspot detection.

The chapter presents a model called Spatio-Temporal Sensor Graph to represent sensor data. Time aggregated graphs are generalized to include probability parameters to incorporate the stochastic nature of sensor graphs.

BASIC CONCEPTS

Traditionally graphs have been extensively used to model spatial networks [24]; weights assigned to nodes and edges are used to encode additional information. For example, the spatial location of a sensor can be represented using the attribute assigned to the node that represents the sensor and the flow rate of a river stream between two sensors can be represented by an attribute of the edge connecting the nodes. In a real-world scenario, it is not uncommon for these parameters to be time-dependent. This section discusses a graph based model that can capture the time-dependence of network parameters. In addition, the model captures the possibility of edges and nodes being absent during certain instants of time.

SPATIO-TEMPORAL SENSOR GRAPH

A graph $G = (N, E)$ consists of a finite set of nodes N and edges E between the nodes in N . If the pair of nodes that determine the edge is ordered, the graph is directed; if it is not, the graph is undirected. In most cases, additional information is attached to the nodes and the edges. In this section, we discuss how the time dependence of these edge/node parameters are handled in the proposed model, the Spatio-Temporal Sensor Graph.

The Spatio-Temporal Sensor Graph is defined as follows:

$$\begin{aligned}
 STSG = (N, E, TF, \\
 f_1 \dots f_k, g_1 \dots g_l, \\
 (nw_1, ne_1) \dots (nw_k, ne_p), \\
 (ew_1, ee_1) \dots (ew_p, ee_p), | \\
 f_i : N \rightarrow \mathbb{R}^{TF}; g_i : E \rightarrow \mathbb{R}^{TF}; \\
 nw_i : N \rightarrow \mathbb{R}^{TF}, ne_i : N \rightarrow \mathbb{PD}, \\
 ew_i : E \rightarrow \mathbb{R}^{TF}, ee_i : E \rightarrow \mathbb{PD})
 \end{aligned}$$

where N is the set of nodes, E is the set of edges, TF is the length of the entire time interval, $f_1 \dots f_k$ are the mappings from nodes to the time series associated with the nodes (e.g., the time instants at which the node is present), $g_1 \dots g_l$ are mappings from edges

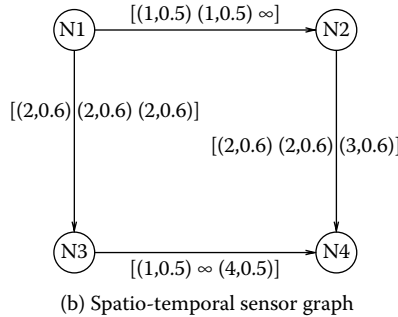
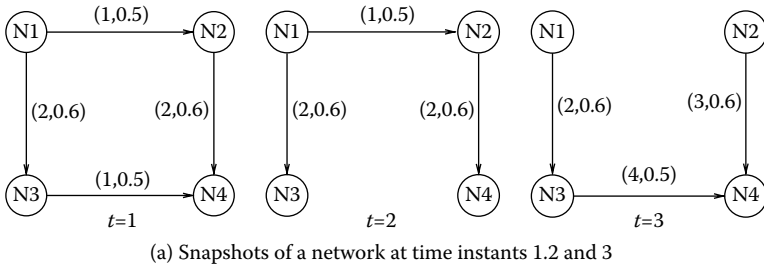


FIGURE 3.3 Spatio-Temporal Sensor Graph and snapshots at various instants.

to the time series associated with the edges, and $(ew_1, ee_1) \dots (ew_p, ee_p)$ indicate the time-dependent attribute on the edges. $\mathbb{P}\mathbb{D}$ indicates a probabilistic error. These attributes are the quantitative descriptors of the physical relationship between the nodes. To represent the stochastic nature of the measured values of physical phenomena, each attribute is a pair that represents the measured value and the associated error.

Example A graph representation of a network at three instants of time is shown in Figure 3.3(a), including temporal changes in connectivity and edge properties (e.g., flow rate). Each edge attribute is a pair \langle measured value, error \rangle . The first parameter in the pair is the measured value and the second is the expected error. For example, the edge from node N3 to node N4 disappears at the time instant $t = 2$ and reappears at $t = 3$, and the attribute on the edge N3–N4 changes from $(1,0.5)$ at $t = 1$, to $(4,0.5)$ at $t = 3$. Figure 3.3(b) shows the Spatio-Temporal Sensor Graph. This is encoded in the time-aggregated graph using the edge attribute series of N3–N4, which is $[(1,0.5), \infty, (4,0.5)]$; the entry “ ∞ ” indicates that the edge is absent at the time instant $t = 2$.

A time-expanded graph would need copies of the entire graph. The number of nodes is larger by a factor of T , where T is the number of time instants and the number of edges is also larger in number compared to the STSG. Typically the value of T is very large in a spatial network such as a sensor graph since the changes in the network are quite frequent. Figure 3.4 shows the time-expanded graph representation

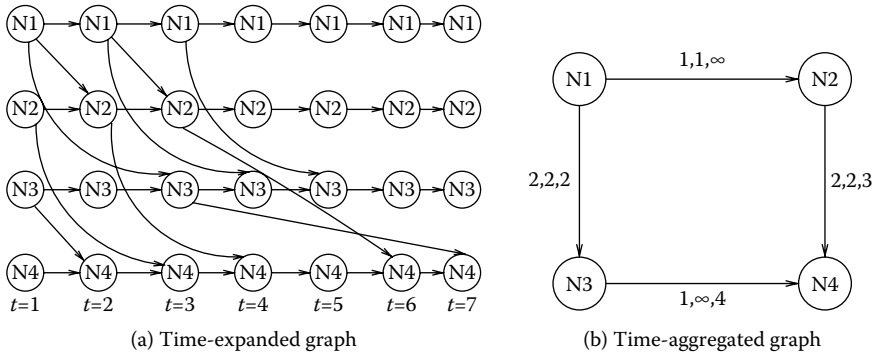


FIGURE 3.4 Time-aggregated graph vs. time-expanded graph.

for the STSG shown in Figure 3.3(b), assuming that the edge attributes are non-probabilistic. This would result in time-expanded networks that are enormously large and would make the computations slow. Though flow-based physical phenomena can be modeled using the cross edges between the copies in a time-expanded graph, it might not be possible to represent every type of physical relationship between the sensors. Moreover, it might not be possible to represent probabilistic edge parameters in a time-expanded graph.

CASE STUDIES

This section presents three case studies using the Spatio-Temporal Sensor Graph (STSG) to discover interesting patterns. First, anomaly detection is presented that generates a set of time series at each sensor node where each time interval is identified as an anomaly. The second case study is the discovery of basic hotspots using the STSG model and the time series that was discovered in the unanticipated anomaly case study. Finally, a case study on the discovery of growing hotspots using the STSG model is presented. The case studies addressing unanticipated anomaly detection, basic hotspot, and growing hotspots are presented in the coming sections. The algorithms presented do not consider the probabilistic parameters in discovering and predicting hotspots. Prediction is done assuming that the physical model of the edge parameter is accurate. However, the algorithms can be extended to incorporate the stochastic nature of the edge parameters.

ANOMALY DETECTION

Definition

The problem of anomaly detection can be generally described as identifying a set of observations that are inconsistent or generated from a different mechanism than the rest of the dataset [24]. In general, anomaly detection can be categorized in five different groups: (a) global anomalies where a data point may be different from the entire dataset and is often used in several traditional aggregate

approaches, (b) spatial anomalies where a data point may be different when comparing against its physical neighbors, (c) temporal anomalies which are similar to spatial anomalies except that the neighborhood region is based on its temporal neighbors, (d) spatio-temporal anomalies where a data point is different based on the neighborhood of both space and time, and (e) network anomalies where a data point is different based on the neighboring connected nodes.

Application

Several interesting applications can utilize anomaly detection such as the monitoring of watersheds and in-plant systems using a sensor network. However, challenges arise in such situations. For example, in a watershed, a level of normality needs to be defined to determine the types of anomalies. Normality may be based on domain models (e.g., physics-based differential equation governing fluid flow, see [5,22] for more information) using some form of an aggregate on historical data gathered by the sensor network. The historical dataset may be broken into several slices (or by month) based on the seasonality. Another challenge includes change detection between sensors in a watershed. For example, suppose we have sensors in place within the length of a river with a train track running along the side of it. Suppose a train carrying harmful contaminants causes a spill into the river stream. The sensors within this part of the river will start observing measurements that are significantly different from domain models as well as in the upstream section of the river. Thus, an anomaly can be detected to warn the water treatment plant.

Method

Algorithm 1 presents the pseudocode to detect spatio-temporal anomalies from a sensor network. There are three types of input to this approach. The first is a set of nodes within the sensor network where each node contains information about the measurements gathered at a time interval. The second is a model defined by the domain scientists to determine the predicted output under normal conditions. Third is the maximum number of time intervals recorded within the sensor network. The output of this approach consists of a set of time series for each node where an anomaly was detected. These time series will be needed as inputs for the case studies discussed in “Basic Hotspot Detection” and “Growing Hotspot Detection.”

Algorithm 1 consists of two phases to detect anomalies within a sensor network. In Phase I, a domain science model is used to calculate the predicted output under normal conditions for each node (Lines 2–7 of Algorithm 1). The input is based on the current node and the predicted output is for the successor nodes, based on a domain model [5,22]. The predicted output is stored at the successor node (Line 5 of Algorithm 1). This process is computed for all time intervals for each node in the sensor graph.

Phase II of Algorithm 1 identifies the anomalies based on the domain science model and the actual measurements from the sensor graph (Lines 8–14 of Algorithm 1). Every node is investigated for anomalies except the first node because the predicted output generated by the domain science model is calculated based on its predecessor node (Lines 8 of Algorithm 1). An anomaly is identified when the actual

Algorithm 1 Pseudocode for Anomaly Detection

```

1: Function ANOMALY(set NodeIds, model m, int maxTime)
   {Phase I: Domain Science Model}
2: for each node n ∈ NodeIds do
3:   for time t = 1 to maxTime do
4:     Calculate predicted output o using m for n.measurements at t
5:     successorNode.predicted.t = o
6:   end for
7: end for
   {Phase II: Identify Anomalies}
8: for node n = 1 to NodeIds.size() do
9:   for time t = 1 to maxTime do
10:    if n.measurements ≠ n.predicted at t then
11:      n.anomaly = n.anomaly ∪ t
12:    end if
13:  end for
14: end for

```

measurements at a node and the predicted output for a time interval are different or the difference is greater than some threshold (Line 10 of Algorithm 1). If an anomaly is found, then the time interval t is assigned to a set of time series for the node (Lines 11 of Algorithm 1). This information is intended for the STSG model which will be used in the next two sections.

Execution Trace

Table 3.1 presents an example dataset containing a set of measurements found in a sensor network. The values depicted in this example use a simple prediction model; for further information on differential models, see [5,22]. In this example, we have five nodes running along a river where the node ID increases based on the direction of the water flow. In each node, there are three time intervals and a corresponding measurement at that sensor (e.g., concentration of a chemical in the water).

In Phase I of Algorithm 1, the predicted outputs are calculated for each node having a predecessor node. An example of these predicted outputs is shown in Table 3.1. Notice that there are no predicted values for node 1 because a predecessor node (e.g., node 0) does not exist in the dataset.

TABLE 3.1
Execution Trace of Anomaly Detection Algorithm

Nodes	1			2			3			4			5		
Time Slot	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
Measure	10	20	30	20	50	50	40	50	50	80	100	50	160	200	50
Predicted				20	40	60	40	100	100	80	100	100	160	200	100
Anomaly				{ t_2, t_3 }			{ t_2, t_3 }			{ t_3 }			{ t_3 }		

In Phase II of Algorithm 1, the anomalies are identified by having a set of time series when the predicted values and the actual sensor measurements differ. Table 3.1 gives an example where nodes 2 and 3 have anomalies occurred at $\{t_2, t_3\}$ and nodes 4 and 5 have anomalies at $\{t_3\}$. Figure 3.5(b) gives an illustration of the STSG model with the anomaly time series found in this execution trace.

Computational Complexity

Algorithm 1 has two phases: (Phase 1) the predicted output for each node calculated for all time intervals having the complexity of $O(nT)$ where n is the number of nodes and T is the total number of time intervals; and (Phase 2) the anomalies are detected by examining $(n - 1)$ nodes (the boundary nodes are not checked) for all time intervals and having the complexity of $O[(n - 1)T]$. Thus, the total computational complexity of both phases 1 and 2 is $O[nT + (n - 1)T] = O(nT)$.

BASIC HOTSPOT DETECTION

Definition

The problem of hotspot detection is to discover the sensor nodes that display significant differences between observed values and expected “standard” values.

Application

In application domains such as river systems where chemical levels are constantly monitored, sensors are deployed to detect the changes. In this context, a hotspot is indicated by a sensor reporting an anomaly (as discussed in “Anomaly Detection”). We discuss a method to discover hotspots using the STSG model. The nodes in the STSG represent the sensors. An edge is added between the nodes if and only if there is a physical relationship between the nodes. The presence of a hotspot at a node at various time instants is indicated by a node time series. In addition, the time dependence of the physical relationships modeled by the edges can be represented by time series attributes. Figure 3.5 illustrates the graph model for the sensor graph. For the sake of simplicity, edge attributes are not shown in Figure 3.5. Figure 3.5(a) shows an example network. The nodes that are active at time instants $t = 2$ and $t = 3$ are shown in Figure 3.5(b) and (c). The Spatio-Temporal Sensor Graph representation is shown in Figure 3.5(d). The time series attributes on the nodes indicate the hotspots at various time instants. For example, the time series 2, 3 on the node N2 indicates that the node is a hotspot at $t = 2, 3$.

Method

Given a sensor graph called the source node, the hotspot at any time instant is the set of nodes where an anomaly has been detected at the given time instant. We use a modified breadth-first strategy to find the nodes that indicate the hotspots at any time instant. The pseudocode is provided in Algorithm 2.

The algorithm finds the hotspots using the STSG model. Each node has a time series attribute that encodes the information about the time instants at which the node

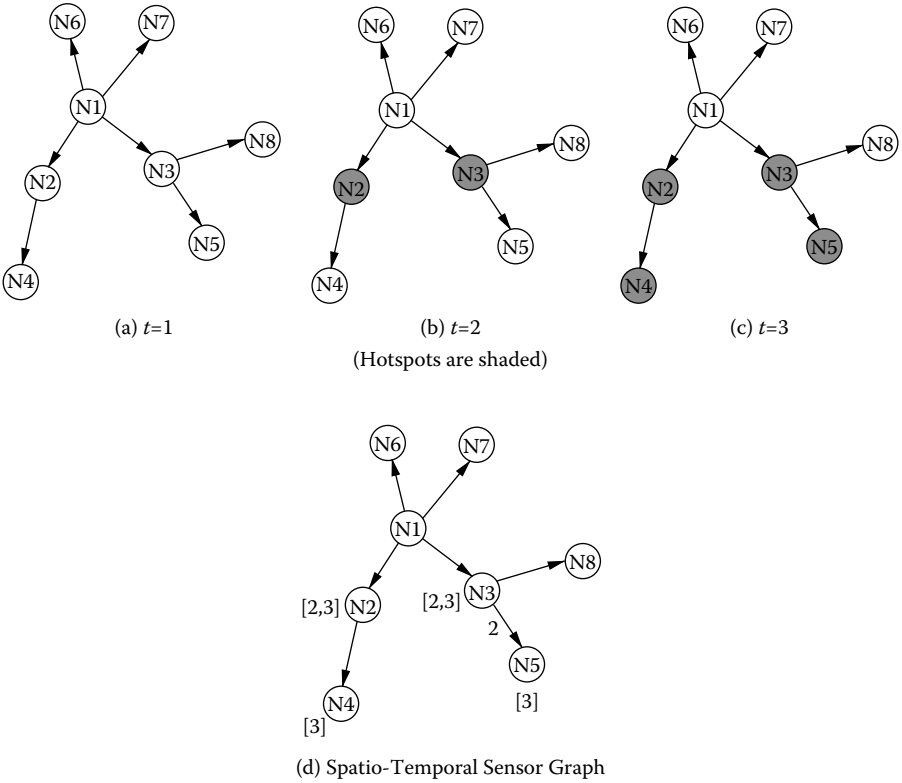


FIGURE 3.5 Spatio-Temporal Sensor Graph model to detect hotspots.

Algorithm 2 Hotspot Algorithm

```

1: Function BASICHOTSPOTS(Graph  $G(N, E)$ , set  $N$ , set  $E$ , node source)
2: for  $t = 1, T$  do
3:   mark source as visited;
4:   enqueue(Q,source);
5:   if  $t$  in node_time_series of source then
6:     hotspots[t] = source;
7:   end if
8:   while Q not empty do
9:      $u = \text{Dequeue}()$ ;
10:    For every node  $v$  such that  $uv \in E$  and if  $t$  in node_time_series
11:    if  $v$  is not marked then
12:      mark  $v$  as visited.
13:      enqueue(Q,v);
14:      hotspots[t] = hotspots[t]  $\cup$   $v$ ;
15:    end if
16:  end while
17: end for

```

has an anomaly. For example, the time series [1,2] at node N2 in Figure 3.5(d) indicates that the node is a hotspot at $t = 2, 3$. The algorithm searches the graph starting at (any) given node for each value of time t and finds the hotspots. The search uses a breadth-first strategy, modified to incorporate the fact that each node has a time series that needs to be checked. When each node is visited, the algorithm checks to see whether it is a hotspot by checking the node time series. The node time series is assumed to be sorted. The output of the algorithm is the set of hotspots at every time instant.

Execution Trace

Table 3.2 shows the trace of the algorithm for the STSG shown in Figure 3.5(d). The search starts at node N1 at $t = 1$ and detects no hotspots. At $t = 2$, the search finds that the nodes N2 and N3 are hotspots based on the entry “2” (indicating the presence of a hotspot at $t = 2$) in their node time series [1,2]. The algorithm performs another iteration for $t = 3$ and finds the hotspots at N2, N3, N4, and N5. The execution trace is summarized in Table 3.2.

Computational Complexity

The algorithm visits every node of the STSG at every time instant t and searches the node time series to detect the hotspots. The algorithm performs $O(n)$ steps to visit all nodes. At each step, the presence of the node at the given time instant is checked. If the time series is sorted, this look-up is $O(\log T)$ since the length of the time series is at most T where T is the length of the time period. At each node this operation has a complexity of $degree(node) \cdot \log T$. The cost over all the nodes is hence $O(m \log T)$ where m is the number of edges in the graph. Since the search is performed at every instant, the computational complexity is $O[(n + m \log T)T]$, where n is the number of nodes, m is the number of edges, and T is the length of the time period.

GROWING HOTSPOT DETECTION

Definition

The problem of growing hotspot detection is to predict expanding patterns that may lead to significant differences between some observations and the rest of the data set. This problem is different from anomaly detection, where a growing pattern may not be identified as an outlier due to the lack of severity at the early stages of growth, for example, slow leakage from buried chemical drums to the ground water supply. Domain knowledge (e.g., flow rate, plume model) is incorporated in Spatio-Temporal Sensor Graph (STSG) representation to predict the growing hotspots.

TABLE 3.2
Execution Trace of the Hotspot Algorithm

Time	$t1$	$t2$	$t3$
Hotspot Nodes	\emptyset	{N2,N3}	{N2,N3,N4,N5}

Application

In application domains such as river systems where chemical levels are constantly monitored, sensors may be deployed to detect the changes. In this context, a growing hotspot is indicated by the increase in the number of sensors reporting an anomaly, over time. We discuss a method to discover growing hotspots using a model called the STSG, which predicts the spread of hotspots. The nodes in the STSG represent the sensors. The edges in the STSG are quantified with the descriptors that represent the propagation of the cause of anomaly. For example, parameters that model the fluid flow between two sensors can be modeled as an edge attribute. Each attribute is a pair that consists of the measured value and the expected error. If the attribute is time-dependent, STSG will represent the attribute as a time series.

Figure 3.6 shows a Spatio-Temporal Sensor Graph that represents a collection of sensors. Nodes represent the sensors and the edges indicate that they are connected (e.g., by fluid flow). The edge parameters represent the characteristics of the underlying connection. In this example, for the sake of simplicity, the edge parameters are shown to be constants, though Spatio-Temporal Sensor Graphs can represent time-varying edge attributes. Figure 3.6(a) shows the state of the sensors at $t = 1$, where the node N1 is active. Figure 3.6(b) and 3.6(c) show the predicted active sensor nodes at $t = 2, 3$. For example, at $t = 2$, node N2 is predicted to be a hotspot and at $t = 3$, N11 is expected to be active. The predicted hotspots are shown as shaded circles in the figure.

Method

Given a set of sensor nodes (called the source nodes), the hotspots at any future time instant can be predicted from the Spatio-Temporal Sensor Graph using the edge attributes that describe the propagation between the sensor nodes. The algorithm finds the nodes that are reachable from the source nodes within the interval between two time instants. These nodes are discovered based on the edge attribute values that quantify the physical relationship between the nodes. Pseudocode of the algorithm is provided in Algorithm 3. Though the model represents each attribute as a pair of values that indicate the measured value and the associated error, this algorithm does not consider the error parameter in its computations. The error values are omitted from Figure 3.6, for the sake of simplicity.

Execution Trace

Table 3.3 gives the trace of the algorithm for a stream network shown in Figure 3.6. For the sake of simplicity, it is assumed that every segment (represented by an edge) has the same length. The edge attributes are the flow rates. The algorithm lists the nodes reachable from every node for each time instant. At $t = 1$, node N1 is a hotspot. Since the flow rate on edge N1-N2 is 1 unit, flow is predicted to reach N2 at $t = 2$ and N2 is added to the list of hotspots at $t = 2$. Similarly, flow is predicted to reach N3 at $t = 2$. Since the rate is faster, the flow moves through N3 and reaches N5 at $t = 2$, adding both N3 and N5 to the list at $t = 2$.

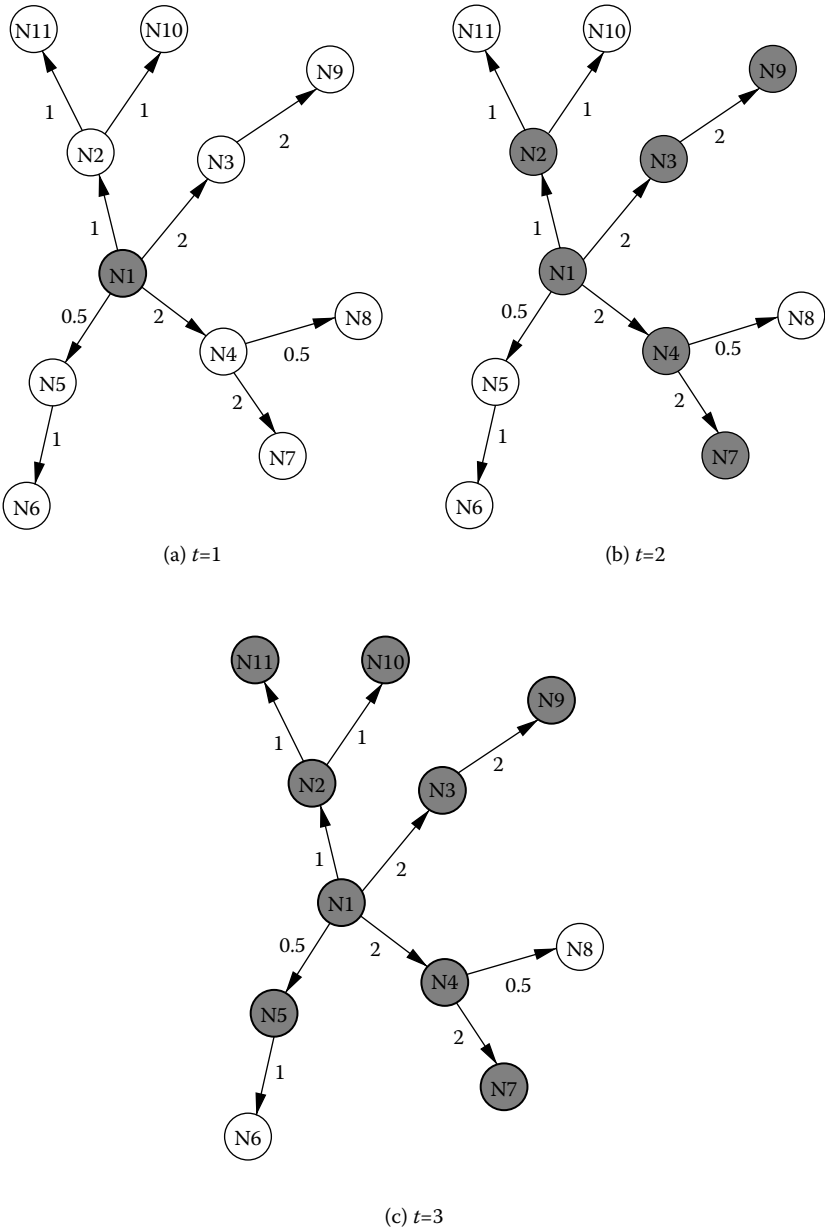


FIGURE 3.6 Detecting growing hotspots.

Algorithm 3 Growing Hotspot Algorithm

```

1: Function SPREADINGHOTSPOTS(Graph  $G(N, E)$ , set  $N$ , set  $E$ , set
    $source\_nodes$ )
2: Enqueue( $Q, S$ );
3: for  $t=1, T$  do
4:    $time\_elapsed[u]= \Delta T$  for all nodes;
5:   while Queue  $Q$  not Empty do
6:      $u = Dequeue()$ ;
7:     For every node  $v$  such that  $uv \in E$ 
8:       if  $v$  is not marked &&  $v$  is reachable based on  $d(u, v)$  then
9:         mark  $v$  as visited.
10:        enqueue( $Q, v$ );
11:        decrement  $time\_elapsed[v]$ ;
12:        Add  $v$  to hotspots[ $t$ ];
13:        return hotspots;
14:       end if
15:     end while
16:   end for

```

Computational Complexity

This algorithm finds the nodes that are reachable from a source node at every time instant. The algorithm performs $O(n)$ steps for each source node. The worst case complexity for each time instant would hence be $O(n^2)$. Since the search is performed for each time step, the computational complexity is $O(n^2T)$, where n is the number of nodes and T is the length of the time period.

CONCLUSION AND FUTURE WORK

The discovery of spatio-temporal patterns in a sensor graph raises several important questions that need to be answered before further analysis. First, how can we model space and time on a sensor graph? Second, can we determine any new or unusual patterns in the sensor graph? Third, which areas in the sensor graph have similar behavior? Finally, how can we predict which nodes in the sensor graph will have similar behavior?

TABLE 3.3
Trace of the Growing Hotspot Algorithm

Time	$t1$	$t2$	$t3$
Predicted	N1	N1,N2,N3,N4	N1,N2,N3,N4,N5
Hotspots		N7,N9	N7,N9,N10,N11

In this chapter, we discuss a Spatio-Temporal Sensor Graph (STSG) model to represent sensor data to answer all of these questions. Three case studies utilizing the STSG model to discover different types of patterns are presented. First, an anomaly detection algorithm that generates a set of time series of where anomalies occur within the STSG is discussed. Second, a basic hotspot discovery algorithm that uses STSG to identify centralized locations at each time interval is described. Third, a growing hotspot method is proposed using STSG to predict nodes that may be hotspots at future time intervals.

The sensor graphs discussed in this chapter emphasized physical networks such as road or river networks. However, the Spatio-Temporal Sensor Graph might be applicable to other types of sensor graphs and this aspect needs investigation. Though the model can incorporate the stochastic nature of sensor graphs, the algorithms currently do not consider the probabilities in computations. Appropriate modifications in the algorithms to account for the stochastic nature of the attributes can be explored.

ACKNOWLEDGMENTS

We are particularly grateful to the members of the Spatial Database Research Group at the University of Minnesota for their helpful comments and valuable discussions. We would also like to express our thanks to Kim Koffolt for improving the readability of this chapter.

This work is supported by the National Science Foundation (NSF-SEI grant), Oak Ridge National Laboratory, Topographical Engineers Corps, and the Minnesota Department of Transportation.

BIOGRAPHIES

Betsy George received the BTech in electrical engineering from the University of Calicut, India and her ME in Computer Science and Engineering from the Indian Institute of Science, Bangalore, India. She is currently working toward her PhD degree in computer science at the University of Minnesota, Minneapolis, MN. Her research interests include spatial databases, spatio-temporal networks, and graph theory. She is a student member of IEEE.

James M. Kang received the BS degree in Computer Science at Purdue University in 2000 and continued to work as a software/systems analyst for Eastman Kodak Company in Rochester, NY. While working he pursued his MS degree in Computer Science at Rochester Institute of Technology. He is currently a doctoral student in the computer science graduate program at the University of Minnesota, Minneapolis, MN. His research interests are in spatial databases, query optimization, spatio-temporal data mining, and applications to Ecology, Civil Engineering, and Geology. He is a student member of IEEE.

Shashi Shekhar received the BTech degree in Computer Science from the Indian Institute of Technology, Kanpur, India, the MS degree in Business Administration and the PhD degree in Computer Science from the University of California, Berkeley, CA.

He is a McKnight Distinguished University Professor at the University of Minnesota, Minneapolis, MN. His research interests include spatial databases, spatial data mining, geographic information systems, and intelligent transportation systems. He is a fellow of the IEEE and a member of the ACM.

REFERENCES

1. A. Arasu, M. Cherniack, E. Galvez, D. Maier, A. S. Maskey, E. Ryvkina, M. Stonebraker, and R. Tibetts. Linear Road: A Stream Data Management Benchmark. *Proceedings of the 30th International Conference on Very Large Data Bases*, Trondheim, Norway, pp. 480–491, Sept. 2005.
2. R. Avnur and J.M. Hellerstein. Eddies: Continuously Adaptive Query Processing. *Proceedings of 19th ACM SIGMOD International Conference on Management of Data*, May, 2000.
3. V. Barnett and T. Lewis. *Outliers in Statistical Data*, John Wiley, New York, 3rd ed. 2003.
4. P. Bonnet, J. Gehrke, and P. Seshadri. Towards Sensor Database Systems. *Lecture Notes in Computer Science*, Vol. 1987, 2001.
5. H. Bravo, J. Gulliver, and M. Hondzo. Development of a commercial code-based two-fluid model for bubble plumes. *Environmental Modelling and Software*. Vol. 22, pp. 536–547, 2007.
6. C. Chong and S.P. Kumar. Sensor Networks: Evolution, Opportunities, and Challenges. *Proceedings of the IEEE*, pp. 1247–1256, Vol. 9(8), 2003.
7. P.S. Corso, M.H. Kramer, K.A Blair, D.G. Addiss, J.P. Davis, and A.C. Haddix. Cost of Illness in the 1993 Waterborne Cryptosporidium Outbreak. *Emerging Infectious Diseases*, Vol. 9(4), pp. 426–431, 2003.
8. D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Next Century Challenges: Scalable Coordination in Sensor Networks. *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, New York, NY, pp. 263–270, 1999.
9. B. George, S. Kim, and S. Shekhar. Spatio-Temporal Network Databases and Routing Algorithms: A Summary of Results. *Proceedings of International Symposium on Spatial and Temporal Databases (SSTD'07)*, July, 2007.
10. B. George and S. Shekhar. Time-aggregated Graphs for Modeling Spatio-Temporal Networks—An Extended Abstract. *Proceedings of Workshops at International Conference on Conceptual Modeling (ER 2006)*, November, 2006.
11. D. Goldin. Spatial Queries over Sensor Networks. *Technical Report*, Department of Computer Science and Engineering, University of Connecticut, March 2003.
12. C. Gotsman and Y. Koren. Distributed Graph Layout for Sensor Networks. *Journal of Graph Algorithms and Applications*, Vol. 9(1), 2005.
13. R. Govindan, J.M. Hellerstein, W. Hong, S. Madden, M. Franklin, and S. Shenker. The Sensor Network as a Database. *Technical Report*, 02-771, Department of Computer Science, University of Southern California, September, 2002.
14. M. Hondzo. Personal communication with Prof. Miki Hondzo, Saint Anthony Falls Laboratory, University of Minnesota.
15. R. Johnson. *Applied Multivariate Statistical Analysis*. Prentice Hall, Upper Saddle River, NJ, 1992.
16. D.E. Kaufman and R.L. Smith. Fastest Paths in Time-Dependent Networks for Intelligent Vehicle Highway Systems Applications. *IVHS Journal*, pp. 1–11, Vol. 1(1), 1993.

17. E. Kohler, K. Langtau, and M. Skutella. Time-Expanded Graphs for Flow-Dependent Transit Times. *Proceedings of the Tenth Annual European Symposium on Algorithms*, pp. 599–611, 2002.
18. K. Lorincz, D.J. Malan, T.R.F. Fulford-Jones, A. Nawoj, A. Clavel, V. Shnayder, G. Mainland, S. Moulton, and M. Welsh. Sensor Networks for Emergency Response: Challenges and Opportunities. *IEEE Pervasive Computing, Special Issue on Pervasive Computing for First Response*, 2004.
19. M. Mauve, J. Widmer, and H. Hartenstein. A Survey on Position-based Routing in Mobile Ad-hoc Networks. *IEEE Network*, Vol. 15, 2001.
20. S. Pallottino and M.G. Scutella. Shortest Path Algorithms in Transportation Models: Classical and Innovative Aspects. *Equilibrium and Advanced Transportation Modelling*, pp. 245–281, Kluwer, 1998.
21. K. Römer, O. Kasten, and F. Mattern. Middleware Challenges for Wireless Sensor Networks. *ACM Mobile Computing and Communication Review*, pp. 59–61, Vol. 6(4), October, 2002.
22. J.C. Rutherford. *River Mixing*. John Wiley and Sons, Inc., New York, NY, 1994.
23. D. Sawitzki. Implicit Maximization of Flows over Time. *Technical Report*, University of Dortmund, 2004.
24. S. Shekhar, S. Chawla. *Spatial Databases: A Tour*. Prentice Hall, Englewood Cliffs, NJ, 2003.
25. S. Shekhar, C.T. Lu, and P. Zhang. Detecting Graph-based Spatial Outliers: Algorithms and Applications. *Proceedings of the 7th ACM International Conference on Knowledge Discovery and Data Mining*, San Francisco, CA, 2001.
26. S. Shekhar, C.T. Lu, and P. Zhang. A Unified Approach to Spatial Outliers Detection. *Geoinformatica, An International Journal on Advances of Computer Science for Geographic Information Systems*, Springer, Netherlands, Vol. 7(2), June, 2003.
27. P. Zhang, Y. Huang, S. Shekhar, and V. Kumar. Exploiting Spatial Autocorrelation to Efficiently Process Correlation-Based Similarity Queries. *Proceedings of the 8th Symposium on Spatial and Temporal Databases*, Santorini Island, Greece, July, 2003.