Discovering Spatio-Temporal Causal Interactions in Traffic Data Streams

Wei Liu
University of Sydney,
Sydney, Australia
wei.liu@sydney.edu.au

Yu Zheng
Microsoft Research Asia,
Beijing, China
yuzheng@microsoft.com

Sanjay Chawla
University of Sydney,
Sydney, Australia
sanjay.chawla@sydney.edu.au

Jing Yuan
University of Science and
Technology of China
yuanjing@mail.ustc.edu.cn

Xing Xie
Microsoft Research Asia,
Beijing, China
xingx@microsoft.com

ABSTRACT

The detection of outliers in spatio-temporal traffic data is an important research problem in the data mining and knowledge discovery community. However to the best of our knowledge, the discovery of relationships, especially causal interactions, among detected traffic outliers has not been investigated before. In this paper we propose algorithms which construct outlier causality trees based on temporal and spatial properties of detected outliers. Frequent substructures of these causality trees reveal not only recurring interactions among spatio-temporal outliers, but potential flaws in the design of existing traffic networks. The effectiveness and strength of our algorithms are validated by experiments on a very large volume of real taxi trajectories in an urban road network.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Data mining

General Terms

Algorithms

Keywords

Spatio-temporal outliers; outlier causalities; frequent substructures; urban computing and planning;

1. INTRODUCTION

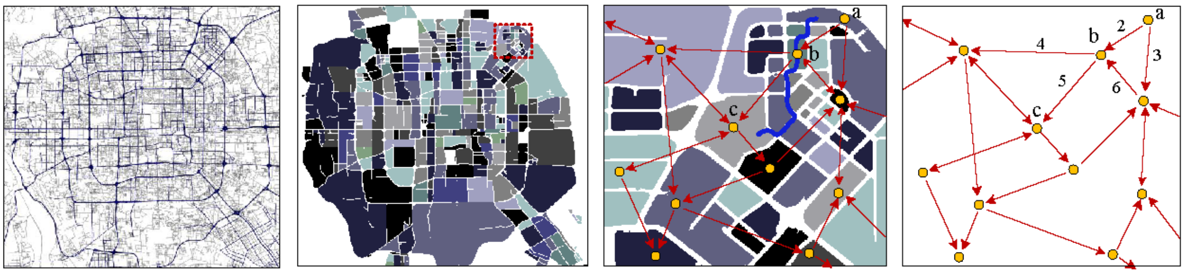
The increasing availability of location-acquisition technologies including GPS and WIFI have resulted in huge volumes of spatio-temporal data, especially in the form of trajectories [2, 5, 7, 15, 14, 19, 17]. Unusual patterns of moving objects' trajectories generally reflect abnormal traffic streams on road networks, which could be caused by aperiodic events including celebrations, parades, large-scale business promotions, protests, traffic control and traffic jams. Therefore, the detection of outliers/anomalies from trajectory data can help in sensing abnormal events and plan for their impact on the smooth flow of traffic. In this study, we treat both known (planned) and unknown (unplanned) events that behave differently from daily network traffics as anomalies.

KDD'11, August 21–24, 2011, San Diego, California, USA.

Challenges and Contributions

In order to successfully detect outliers and causal interactions among them, the following challenges need to be addressed: (i) Heterogeneous traffic patterns: the traffic patterns on roads vary across days of a week and hours of a day. Different road segments have often distinct time-variant traffic patterns. It is difficult to use one model to detect outliers across the road network at different time periods. (ii) Data sparseness and distribution skewness: even though we could have a large number of sensors (taxis) probing the traffic on roads, there are many roads that have only a small number of samples given a large size of road networks in a major city. Moreover, a few road segments are traveled by thousands of vehicles in a few hours, while some segments may be only driven on a few times in a day. These two properties together result in unique challenges in detecting outliers from traffic data. (iii) Causality among outliers: we not only need to discover outliers from the traffic, but also infer causal relationships and interactions among them, especially given the large number of outliers that could be identified. So a challenge is how to detect the appearance, growth, disappearance and transformation of outliers by time (e.g., propagation of a traffic jam).

In this paper we design several steps to address the above challenges and propose solutions to the problem of detecting spatio-temporal outliers and causal relationships among them from traffic data streams. We use contexts of road networks in this study, however, algorithms proposed in this paper can be generally applied to domains of networking [12,



(a) Road network of Beijing (b) Partitioned regions (c)Example of traffic among regions (d) A graph of regions

Figure 1: An example using traffic networks in the city of Beijing. Based on major roads in the traffic network, the entire city (subfigure (a)) is partitioned into regions (subfigure (b)). Trajectories of moving objects (such as a moving taxi shown by a blue trajectory in subfigure (c)) connect neighboring regions, based on which we create the notion of links (subfigure (d)).

13] and climate change [18, 23, 8] etc. More specifically, the **contributions** we make in this paper are:

1. *City-wide traffic modeling*: we partition the urban area of a city into regions using the framework of the city's road network. Then we build a region graph where a node is a region and a link captures the traffic flow among two regions. We formulate the outlier detection problem as identifying the most outlying "links" from the region graph in terms of the spatio-temporal properties of a link.

2. *Outlier tree construction*: we propose the `STOTree` algorithm based on both spatial and temporal properties of detected outliers (which are certain "links" in a time frame) to construct outlier trees, which uncover causal relationships among these outliers.

3. *Frequent outlier subtree discovery*: we propose the `freqentSubtree` algorithm, inspired by association rule mining, which generates the most frequent sub-structure (subtree) from all discovered outlier trees. These frequent subtrees reveal recurrent abnormalities in the data and suggest inherent problems in existing road networks.

The rest of the paper is structured as follows. In Section 3 we introduce the overall framework of our model, including preliminary concepts and notations that we use in this paper. In Section 4 we propose algorithms for discovering spatio-temporal outliers and causal relationships. Experiments and their analysis are reported in Section 5. We conclude in Section 6 with directions for future work.

2. RELATED WORK

To the best of our knowledge this is the first paper that proposes the problem of discovering casual relationships among spatio-temporal outliers. However, there have been a number of efforts on detecting only outliers from spatially and temporally distributed data. For example, principle component analysis (PCA) has been used for network-wide anomaly detection [13, 26, 20, 1]. However, PCA results (as we will show) cannot capture volume heterogeneity and are also very sensitive to parameter settings which are highly data dependent. Under certain circumstances, large anomalies in turn can effect the PCA computation leading to both false positives and false negatives [20].

The problem of outlier monitoring has also been studied in [2] which builds local clusters on trajectory streams and monitors outliers that are defined by a "trajectory" (instead of a spatial link as ours). Another method is Sun *et al.* [22] where a measure, spatial local outlier measure (SLOM), is proposed to capture the local behavior of datum in their spatial neighborhood. This measure takes into account the local stability around a data point and suppresses the reporting of outliers in highly unstable areas. A generalized local statistical (GLS) framework is proposed in [6] which studies the performance of local based methods on detecting outliers in geo-statistical data with either linear or nonlinear trends, and compares them against global based methods. Wu *et al.* [23] design algorithms detecting the most abnormal discrepancy regions in precipitation data, where they use four sweep lines to form grids which are treated as regions. However, none of these approaches model and capture temporal relations (causalities) among detected outlying regions. Lee *et al.* [15, 14] have designed a "partition-and-group" framework for clustering and detecting trajectory outliers. In their approach, they first partition the trajectories into small segments and then use both distance and density to detect abnormal sub-trajectories. This is different from our work as we detect abnormal regions and links of the entire traffic network (instead of objects moving in the network).

Moving objects are usually associated with periodic behavioral patterns, and there have been several methods proposed to address the problem of detecting such periodic movements [3, 4, 10, 9, 16]. Cao *et al.* [3, 4] proposed abbreviated list tables (ALT) to find subsequences that appear periodically and frequently in data sequences, but the periodic patterns they detect are very sensitive to parameter settings. Similarly, Elfeky *et. al.* [9] have proposed specific definitions of periodicities and algorithms for identifying the periodic patterns.

3. OVERVIEW

In this section, we introduce our notations, definitions and the main structure of the proposed model.

3.1 Preliminary Concepts

The overall traffic map is partitioned into regions (Rgn) bounded by high level (i.e. major) roads, each of which may consist of a number of road segments. Figure 1(a) and 1(b) demonstrate an example of region formations.

DEFINITION 1. Trajectory*: A trajectory* Tr *is a trace cre-*

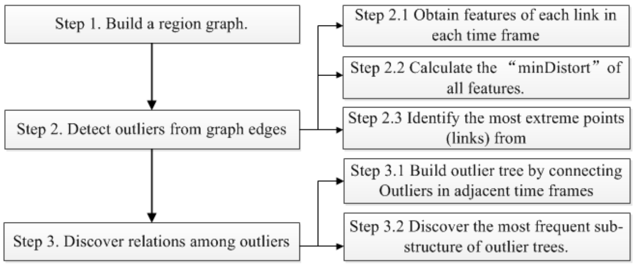


Figure 2: The overall structure of our model for detecting spatio-temporal outliers and their inter-causalities.

ated by a moving object in geographical space. A Tr is represented by of a set of time-ordered points, e.g. $Tr : p_1 \rightarrow p_2 \rightarrow ... \rightarrow p_n$, where each point consists of a geospatial coordinate set and a timestamp, i.e. p = (longitude, latitude, timestamp).

DEFINITION 2. Transition*: Given a trajectory $Tr : p_1 \rightarrow p_2 \rightarrow ... \rightarrow p_n$, there exists a transition S between Rgn_1 and Rgn_2, if there exists adjacent points p_i and p_{i+1} ($1 \leq i \leq n + 1$) such that p_i is in Rgn_1 and p_{i+1} is in Rgn_2, and Rgn_1 is not the same to Rgn_2. A transition is associated with a leaving time (timestamp of p_i) and an arriving time (timestamp of p_{i+1}).*

DEFINITION 3. Link*: A link (Lnk) is comprised of a pair of regions (<Rgn^o, Rgn^d>) indicating a virtual spatial connection between the origin region and the destination region. There exists a link from one region (Rgn^o) to another (Rgn^d) if and only if there exists at least one object moving from Rgn^o at timestamp ts_i to Rgn^d at ts_{i+1}. Figure 1(c) and 1(d) give examples of links.*

DEFINITION 4. Time frame*: A time frame (tf) is a set of consecutive time intervals*[1]*. Figure 3(a) shows an example of a time frame.*

DEFINITION 5. Spatio-temporal outlier*: A spatio- temporal outlier (STO) is a link whose non-spatial and non-temporal attribute values are very different from the values of its spatio-temporal neighbors. Spatio-temporal neighbors of a link Lnk_i are the links whose locations and timestamps are close to those of Lnk_i.*

DEFINITION 6. Outlier causality*: STO_2 (with a region pair <Rgn_2^o, Rgn_2^d> and a time frame tf_2) is caused by STO_1 (containing a region pair <Rgn_1^o, Rgn_1^d> and a time frame tf_1) if and only if the following conditions hold true: (*i*) The destination of STO_1 is the same as the origin of STO_2 (i.e. $Rgn_1^d = Rgn_2^o$); (*ii*)Time frames tf_1 and tf_2 are consecutive to each other and tf_1 is ahead of tf_2.*

During the construction of an outlier tree, an outlier STO_i is a a child of another outlier STO_j if STO_i is caused by STO_j.

3.2 Framework

The main structure of our model is illustrated in Figure 2. The three main steps are preprocessing traffic data to build a region graph, detecting outliers and finally discover causal relationships between the discovered outliers. The second and the third step have three and two sub-steps respectively. Details of these (sub-)steps are described in the following section.

[1]We call a "time interval" a "timebin" in the rest of the paper.

4. METHODOLOGY

In this section, we provide details of our model as shown in Figure 2. Specifically, we focus on the detection of spatio-temporal outliers based on each link's "distort", construct outlier causality trees based on these outliers, and discover the most frequent causal trees which are indicative of recurrent traffic abnormalities.

4.1 Building the Graph of Regions

In our study, we assume the map of traffic network, the set of major roads, and the trajectories of objects are all known. Although it is more straightforward to apply a simple "$n \times m$" grid on maps to define regions, cells of a grid are equal-sized and do not reflect natural differences of regions in a traffic network. So instead of using equal-sized grids, we define regions of a traffic network by road segments as illustrated in Figure 1. In detail, we build a graph of regions according to the following three steps.

1. *Region Partitioning*: As shown in Figure 1(a) and Figure 1(b), we partition a city into dis-jointed regions using the major roads of the city. Here we employ Connected Components Labeling (an image segment method) [21] to partition a map into regions effectively and efficiently, since the problem of subdivisions in a polygonal region is known to be NP-complete [11].

2. *Formulating transitions*: By scanning the trajectory data set, we transfer each trajectory into a sequence of transitions between pairs of regions in terms of definition 2. As demonstrated in Figure 1(c), a trajectory passing three regions a, b, and c results in two transitions: $a \Rightarrow b$, and $b \Rightarrow c$.

3. *Generating links*: If there is a transition generated between two regions, we connect the two regions with a link (refer to Definition 3). In timebin j, a link i (Lnk_i = <Rgn^o, Rgn^d>) is associated with a feature vector of three properties $\vec{f}_{i,j} \equiv$ <$\#Obj$, Pct_o, Pct_d>:

   (a) $\#Obj$: Total number of objects on the links (i.e. objects moving from Rgn^o to Rgn^d in this timebin);

   (b) Pct_o: The proportion of $\#Obj$ among all objects moving out of Rgn^o in this timebin;

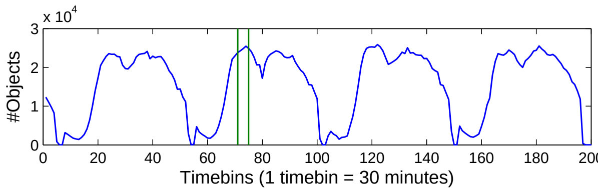
   (c) Pct_d: The proportion of $\#Obj$ among all objects moving into Rgn^d in this timebin;

   Then, using Figure 1(d) as an example (where the number shown on each link is the number of transitions pertaining to the link), the property of link $a \Rightarrow b$ is $\vec{f}_{i,j}$ = <$\#Obj$=2, $Pct_o = \frac{2}{2+3}$=0.4, $Pct_d = \frac{2}{2+6}$=0.25>.

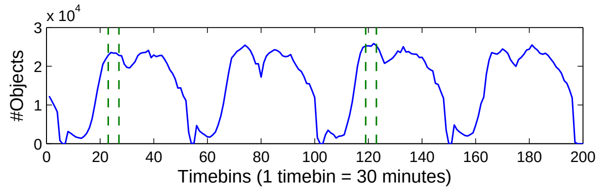
4.2 Detecting Outliers from Graph Links

Assume each time frame is comprised of a fixed number of q timebins. Given a time frame tf_j, we denote the sequence of feature values of a link Lnk_i in this time frame by:

$$F_{i,j} = < \vec{f}_{i,j-q+1}, \vec{f}_{i,j-q+2}, ..., \vec{f}_{i,j} > . \quad (1)$$



(a) An example of time frame defined by the segment between two vertical (green) lines.



(b) Time frames to be compared with the one in (a)

Figure 3: An example of the number of taxis on a certain link in four adjacent days. Gaps between green vertical lines represent time frames, each of which is comprised of several timebins. One unit of timebin in x-axis represents 30 minutes (i.e. 48 timebins comprise a day). The value of *minDistort* of the time frame in subfigure (a) is obtained by calculating the smallest difference between the time frame in (a) and the ones at the same time in adjacent days as shown in subfigure (b).

For each link (Lnk_i) in each time frame tf_j, we calculate the distortion between two time frames (denoted by $minDistort_{i,j}$) by searching for the minimum difference between tf_j and the same time frames of the same days on consecutive weeks. With this approach *minDistort* is capable of capturing the special pattern of traffic data that similar behaviors are observed during the same time of different days or the same day of different weeks etc.

Algorithm 1 shows the procedure of calculating distorts. In line 7 of the algorithm, we obtain the difference between two time frames of a link by computing their Euclidean distance:

$$Distance(tf_j, tf_t, Lnk_i) = \sqrt{\sum_{k=0}^{q-1} \|\vec{f}_{i,j-k} - \vec{f}_{i,t-k}\|^2} \qquad (2)$$

We use $minDistort_{i,j}$ obtained from Algorithm 1 as the "non-spatial and non-temporal attributes" (see Definition 5) of each link in each time frame. Extreme values among *minDistort* of all links are identified as temporal outliers. By subtracting the *min* and dividing by the *max* the feature values of the links are in the range of [0,1]. The normalization removes the effect of different regions and volume sizes. Another advantage of using *minDistort* is that it prevents the examination of many repeating patterns (where $minDistort \approx 0$).

Then *for each time frame*, there is a corresponding three dimensional vector (formed by features <#Obj, Pct$_o$, Pct$_d$>) shown in Figure 4. As each point represents a link, we identify the most extreme points as outliers in that time frame. To normalize the effect of variances along different directions, we use the Mahalanobis distance (instead of Euclidean distance) to measure the the extremeness of data points. We

Algorithm 1 `minDistort`: calculating minimum distort of time sequences

Input: Lnk_i: a link; tf_j: a time frame; t: number of adjacent weeks to check
Output: $minDistort_{i,j}$: the degree of distort for link Lnk_i in time frame tf_j

1: $minDist \leftarrow$ +Infinity;
2: $T \leftarrow tf_{j \pm u\ weeks,\ u \in \{-t,...,t\}}$
3: **for** All time frames tf_t in T **do**
4:   **if** tf_t overlap with tf_j **then**
5:     Continue;
6:   **end if**
7:   $currentDist \leftarrow$ Distance(tf_j, tf_t, Lnk_i);
8:   **if** $currentDist < minDist$ **then**
9:     $minDist \leftarrow currentDist$;
10:   **end if**
11: **end for**
12: Return $minDist$;

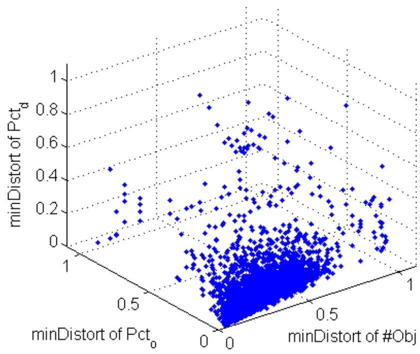


Figure 4: An illustration of the three-dimensional unit cube formed by three features <#Obj, Pct$_o$, Pct$_d$> before normalization for computing the Mahalanobis distance. We label the most "extreme" points among all points as outliers. For example, outlier points are the ones whose distance are the farthest to the center of the data cluster.

use the Mahalanobis distance here in order to normalize the distance by the variance in different directions.

In this way, **the outliers detected are links whose features have the largest difference from both their temporal neighbors (for using "*minDistort*") and spatial neighbors (for being detected "among all links")** – so they are *spatio-temporal* outliers (STOs). Another advantage of identifying "extreme points" as outliers is that it can detect abnormal links with either too low volumes or too high volumes since extremeness of points are based on their Mahalanobis distances.

Now each STO is a spatial **link** associated with a time frame. We represent a STO by its link Lnk_i (containing an original region and a destination region) and its time frame tf_j, i.e., $\text{STO}_{i,j} = < Rgn_{i,o}, Rgn_{i,d}, tf_j >$.

4.3 Constructing Outlier Trees

We propose an algorithm named `STOTree` that finds outlier dependencies by looking at the relationship of outliers from the earliest time frame through the last. **The main insight of `STOTree` is that an outlier STO_1 is a parent of another outlier STO_2 if STO_1 occurred before**

Algorithm 2 `STOTree`: constructing all outlier trees

Input: $STOutlier$: a set of spatial-temporal outliers of size $t \times k$ where t is the number of time frames, and k is the number of outliers to examine in a time frame.
Output: $STOTrees$: a list of roots of spatial-temporal trees.

1: $STOTrees \leftarrow$ an empty set {};
2: **for** Each time frame i ($i \in (1, ..., t)$) **do**
3:   **for** Each outlier j ($j \in (1, ..., k)$) in time frame i **do**
4:     $STORoot_{i,j} \leftarrow$ `FindAllChildren`($STOutlier_{i,j}$, i);
5:     $STOTrees \leftarrow STOTrees \cup STORoot_{i,j}$;
6:   **end for**
7: **end for**
8: Return $STOTrees$;

Subroutine: `FindAllChildren`($STOutlier_{i,j}$, i)
9: **if** Time frame i is the last time frame **then**
10:   Return $STOutlier_{i,j}$;
11: **end if**
12: $STOutlier_{i,j}.subnodes \leftarrow$ an empty set{};
13: **for** Each outlier u ($u \in (1, ..., k)$) in time frame $i + 1$ **do**
14:   **if** $STOTrees$ contains $STOutlier_{i+1,u}$ **then**
15:     continue;
16:   **end if**
17:   **if** $STOutlier_{i,j}.Rgn^d == STOutlier_{i+1,u}.Rgn^o$ **then**
18:     $STOutlier_{i,j}.subnodes \leftarrow STOutlier_{i,j}.subnodes \cup FindAllChildren(STOutlier_{i+1,u}, i + 1)$;
19:   **end if**
20: **end for**
21: Return $STOutlier_{i,j}$;

STO_2 **in time and they are spatially correlated.** Algorithm 2 demonstrates the process of constructing outlier trees from discovered outliers. Note the algorithm results in a collection of trees (a forest). The subroutine (Line 9 to 21) is a recursive function used to retrieve all possible descendants of a node. For each time frame, this recursive function is called on each outlier of the current time frame to compare with each outlier of next time frame, unless the "current" outlier tree already contains outliers of next time frame (Line 14 to 16). So the overall time complexity of the outlier tree construction process on each time frame is upper bounded by O(k^2), where k is the number of outliers in a time frame.

We do not place a restriction on the maximum size of outlier trees in the `STOTree` algorithm, under the assumption that abnormal events caused by one single accident are not expected to last for a long time and the size outlier trees should not grow infinitely. In Section 5.3 we provide empirical evidence that confirms the maximum size of trees is usually small.

Now we give an example by using Figure 5 to demonstrate the process of Algorithm 2 for building outlier trees. Figure 5 uses top 3 outliers in three consecutive time frames, so the input parameters in Algorithm 2 in this case are $k = 3$ and $t = 3$. The algorithm starts from time frame 1 (Line 2 of Algorithm 2), and for each of the top three outlying links (Line 3 to 6), i.e., $A \Rightarrow B$, $C \Rightarrow D$ and $E \Rightarrow F$, the algorithm searches in time frame 2 (Line 13 to 20) and checks whether there is any following link that can be a child of a previous link (Line 17 to 19). This allows the algorithm to find outlying links $B \Rightarrow G$ and $B \Rightarrow E$ as children of $A \Rightarrow B$; and similarly it identifies link $H \Rightarrow K$ in time frame 3 as a child of $J \Rightarrow H$ in time frame 2. Therefore two outlier trees are built up as shown in the right side of Figure 5. In this

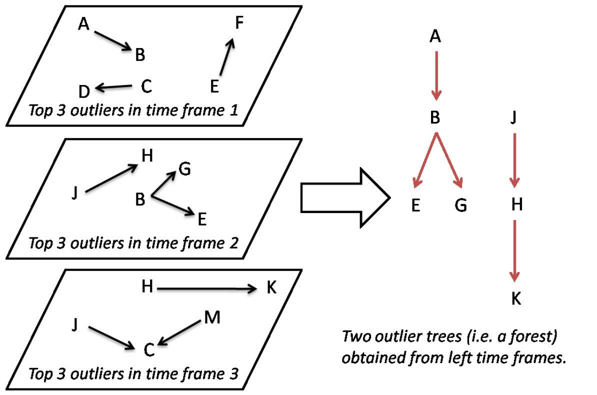


Figure 5: A synthetic example for demonstrating the process of building a forest of two outlier trees. The subfigure on the left illustrates top 3 outlying links in three consecutive time frames, and the one on the right shows two outlier trees obtained from these time frames.

way, Algorithm 2 scans through all time frames of traffic data we have, and builds a forest of various outlier trees.

4.4 Causal Outlier Detection

Denote by T the forest containing all outlier trees. The most significant and recurring causal relationships correspond to the most frequent subtrees of T. The mechanism of discovering frequent subtrees from all outlier trees is inspired by the process of mining frequent item sets, except that we design our own strategy to generate frequent subtree candidates (through node insertion on trees).

The process of discovering frequent substructures from constructed outlier trees is shown in Algorithm 3. Given a predefined support threshold ϵ, we first find all single nodes whose supports exceed ϵ (Line 3 of the algorithm), then we use this set of frequent single nodes to form candidates of frequent subtrees. The "while" iteration (Line 6 to 30) first generates candidates of subtrees (Line 9 to 15), and then checks the support of each candidate and performs filtering (Line 18 to 29) according to ϵ.

When generating subtree candidates, new subtrees (whose sizes are increased by one) are created by inserting a frequent single node into previous frequent subtrees. This node insertion process is given in Algorithm 4. The single node to be inserted is first compared with the root of the tree, and is inserted as a subnode of the root (Line 1 to 3 of Algorithm 4) if the root can be a parent of the single node and its existing children do not contain the single node. Otherwise, the single node is compared and checked whether it can be inserted into branches below the root (i.e. a recursive process shown in Line 8 to 12). When counting the support of a candidate subtree, we increase the frequency of the candidate by one if all nodes (with their immediate subnodes) of the candidate have an exact match with a discovered outlier tree (Line 21 to 23).

The effectiveness and strengths of our algorithms, `STOTree` and `frequentSubtree`, are evaluated in the next section.

5. EXPERIMENTS AND ANALYSIS

In this section we report on the experiments carried out on taxi trajectory data on the road network of Beijing city.

Algorithm 3 `frequentSubtree`: discovering frequent subtrees from STOutlier trees

Input: $STOTrees$: a list of roots of spatial-temporal trees; ϵ: a support threshold for frequent substructure selection.
Output: $freqentSubtrees$: a list of roots of frequent spatial-temporal subtrees.

// Form a list of frequent nodes (i.e. frequent trees of size 1).
$numTrees \leftarrow$ number of roots in $STOTrees$;
$frequentNodes \leftarrow$ unique nodes appearing at least $numTrees \times \epsilon$ times in $STOTrees$.
$mergeTarget \leftarrow frequentNodes$
$freqentSubtrees \leftarrow$ an empty set{};
while size($mergeTarge$) > 0 **do**
  // Form candidates of frequent subtrees;
  $subtreeCandidates \leftarrow$ an empty set {};
  for Each node $singleton_i$ in $mergeTarge$ **do**
    for Each root $root_j$ in $mergeTarget$ **do**
      if `nodeInsertion`($root_j$, $singleton_i$) **then**
        $subtreeCandidates \leftarrow subtreeCandidates \cup root_j$;
      end if
    end for
  end for
  // Filer subtree candidates by threshold of support ϵ;
  Clear $mergeTarget$;
  for Each candidate $candidate_i$ in $subtreeCandidates$ **do**
    $count \leftarrow 0$;
    for Each root $root_j$ in $mergeTarget$ **do**
      if $root_j$ contains $candidate_i$ **then**
        $count \leftarrow count + 1$;
      end if
    end for
    if $count > \epsilon \times numTrees$ **then**
      $freqentSubtrees \leftarrow freqentTrees \cup candidate_i$;
      $mergeTarget \leftarrow mergeTarget \cup candidate_i$;
    end if
  end for
end while
Return $freqentSubtrees$;

Table 1: Statistics of regions and links in the graph built from road network traffics.

#Regions	#Links	Avg. #Obj	Avg. Pct_o	Avg. Pct_d
396	10109	742.48	27.07	27.7702

Our experiments are conducted on a 64 bit server with 3.2 GHz CPU and 8 GB memory. We note that although we are using road traffic data, our methods and algorithms can be easily adapted into other domains such as finding anomalies in the internet traffic data and even climate data.

5.1 Data and Parameters

Data: We test our algorithms based on a real GPS trajectory dataset generated by 33,000 taxis in a period of 6 months (from 01/03/2009 to 31/08/2009) [25, 24] . The total distance traveled by the taxis is more than 800 million kilometers and the total number of GPS points is nearly 1.5 billion. The average sampling interval and average distance of between two consecutive points are around 3.1 minutes and 300 meters respectively.

We define ten minutes as a timebin, and six timebins as a time frame (hence one time frame represents an hour). We check two adjacent weeks in $minDistort$ calculations (i.e. the parameter t in Algorithm 1).

Road Network: We perform evaluations based on the road network of Beijing, which contains 106,579 road nodes and 141,380 road segments.

Algorithm 4 `nodeInsertion`: inserting a node to an outlier tree

Input: $Root$: a root of an outlier tree; $Singleton$: a node to be inserted.
Output: $true/false$: whether or not the node insertion is successful.

if $Root.Rgn_d$ equals $singlton.Rng_o$ && $Root.subnodes$ does not contain $singlton$ **then**
  $Root.subnodes \leftarrow Root.subnodes \cup singlton$;
  Return $true$;
else
  if size($Root.subnodes$) == 0) **then**
    Return $false$;
  else
    for Root of each subnode $subRoot$ in $Root.subnodes$ **do**
      if $InsertNode(subRoot, Singleton)$ **then**
        Return $true$;
      end if
    end for
  end if
end if
Return $false$;

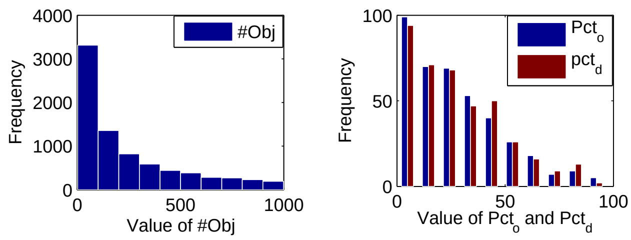


Figure 6: Histograms of the three features (i.e. <$\#Obj$, Pct_o, Pct_d>) of all links in the region graph we obtained.

5.2 Results from Building Graphs

The statistics (e.g. number of regions and links) of the graph built are presented in Table 1. We note that in our evaluations, the number of links built by using 1 month, 2 months, ..., until all 6 months of taxi trajectories remains constant at 10109. This observation implies that although the links are *dynamic*[2] (dependent of traffic data), the model we have built does not miss any links even if one uses only a small fraction of taxi trajectories to build the underlying traffic graph. The distribution of the three features of links are presented in Figure 6.

5.3 Evaluations on Outlier Trees

In this experiment, we bound the value of k (i.e. number of outliers to be identified in each time frame) between 1 and 99, and report its effects on constructing outlier trees (shown as in Figure 7 and 8).

We set the minimum size of a tree (i.e. total number of nodes) to 2, and hence ignore singles nodes (trees of size 1) in counting final outlier trees. What we observe from Figure 7 is that, although the total number of trees constructed by detected outliers increase substantially when k increases (left subfigure), the maximum size of all trees has

[2] By contrast, regions are always *static* (independent of traffic data).

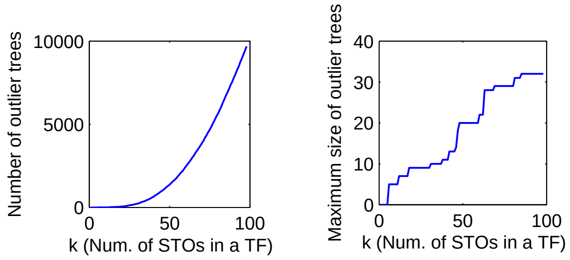


Figure 7: Total number of outlier trees and the maximum size of trees under different settings of k (the number of outliers in a time frame to construct outlier trees). "TF" in the label of x-axis is short for "time frame". While the number of trees increases dramatically when k increases, the maximum size of trees grows relatively smoothly.

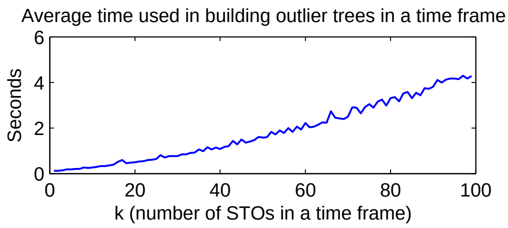


Figure 8: Average time (in seconds) used for constructing outlier trees under different settings of k (the number of outliers in a time frame to construct outlier trees). When k increases, the average time grows almost linearly.

a smoothly increasing trend (right subfigure). This observation validates our earlier belief that abnormal events caused by one single accident normally do not last very long, and that the maximum size of trees is usually small.

In Section 4.3 we have explained that the time complexity of our outlier tree construction algorithm `STOTree` is in the worst case (i.e. upper bounded by) $O(k^2)$. Here we present empirical results that illustrate `STOTree` runs much faster than $O(k^2)$ in practice. From Figure 8 we can observe that the average time used for building trees in a time frame increases almost linearly (instead of quadratically) with k. This indicates `STOTree` can potentially be used in an online setting, and can detect outlier causalities on the fly.

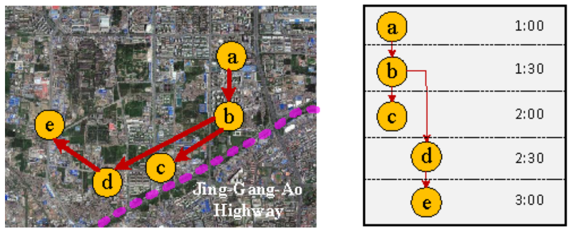
Moreover, the spatio-temporal outliers and their causal interactions detected by our algorithms all coincide with known abnormal events. The following are two prominent examples of known events:

Known event 1. The Bejing–HongKong highway was under traffic control for building viaducts of a Beijing light rail (i.e. the light rail of Fangshan line[3]) at midnight of the May 5th, 2009;

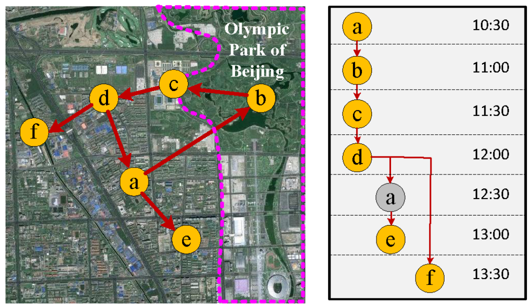
Known event 2. The entry fee into Olympic sports center was waived during day time on 7th August, 2009.

By using our model, the above two known outliers are successfully detected as shown in Figure 9(a) and 9(b) respectively. However, they are not detected by PCA based techniques as in [13]. As Figure 10 shows, the links of these two known abnormal events are among other points and are

[3]`http://en.wikipedia.org/wiki/Fangshan_Line,_Beijing_Subway`



(a) An outlier tree besides Beijing–HongKong (Jing-Gang-Ao) highway between 4th and 5th ring road in time frames of 1:00am to 4:00am, on May 5th, 2009.



(b) An outlier tree around Olympic sports center of time frames from 10:00am to 2:30pm, on August 07, 2009. Note that there exists a loop in this tree, where region a firstly showed up as an origin of an outlying link $a \Rightarrow b$, and then later became a destination of another outlying link $d \Rightarrow a$.

Figure 9: Examples of two outlier trees. These outlier trees coincide with the events that (a) Bejing–HongKong highway was under traffic control for constructing viaducts in the midnight of May 5th, 2009; (b) the Olympic sports center was free of charge to visit at August 7th, 2009.

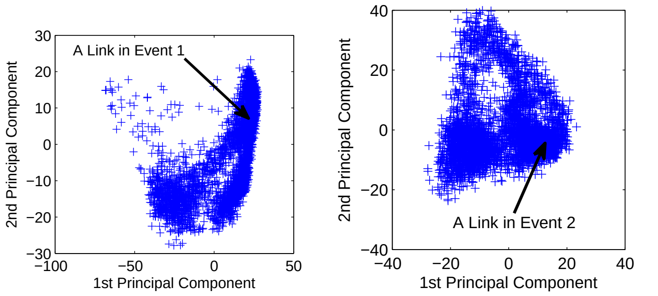
undistinguishable in the coordinate framework formed by the first two principle components from PCA.

A major reason for the failure of PCA is that the first known outlier (event *1*) occurred in the *off-peak hour* of most links, and the second known outlier (event *2*) happened in the *peak hour* of most links – in other words, they are captured by the smallest principle components (PCs) and the largest PCs respectively. The difficulty of choosing appropriate PCs makes it hard to use PCA to detect both of these two traffic events.

5.4 Evaluations on Frequent Subtrees

In this experiment, we control the value of support threshold ϵ, and report its effects on discovering frequent outlier subtrees in terms of the number of subtrees passing through ϵ and their maximum support. We test the properties of frequent subtrees on five sets of support threshold: $\epsilon \in \{0.01, 0.05, 0.1, 0.15, 0.2\}$. We ignore values of ϵ higher than 0.2, since the total number of frequent subtrees is already considerably small when $\epsilon = 0.2$ (as shown in Figure 11(a)). However, as long as the number of subtrees is higher than zero, the `freqentSubtree` algorithm can still identify the most frequent subtree that has the highest support (as shown in Figure 11(b) on $\epsilon = 0.15$). This observation illustrates the *completeness* of the frequent subtrees generated by the `freqentSubtree` algorithm.

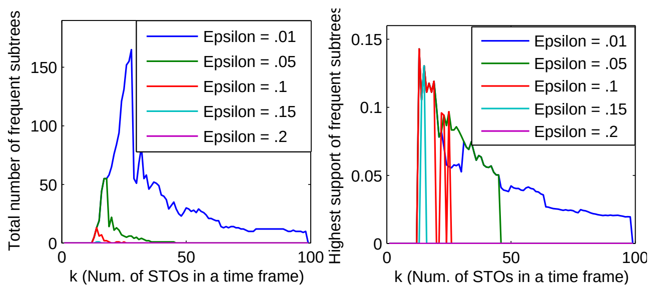
Recall that regions indicated by the most frequent (i.e.



(a) Coordinate formed by top two principle components in time frames of known event 1.

(b) Coordinate formed by top two principle components in time frames of known event 2.

Figure 10: Data points in terms of links in the coordinate formed by top two principle components from PCA. The two known abnormal events detected by our model can not be identified by PCA approaches.



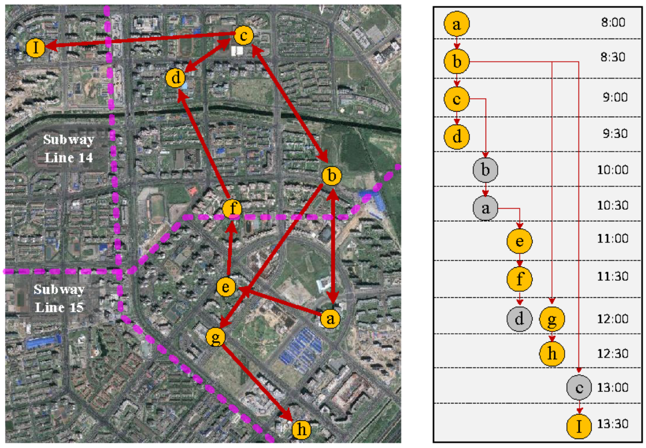
(a) (b)

Figure 11: Total number of frequent subtrees and the highest support among them under different settings of k and support threshold ϵ. The number of discovered trees becomes significantly small when the support threshold is high (e.g. $\epsilon = 0.2$ in subfigure (a)). However, as long as the number of subtrees is higher than zero, the frequent subtrees that have the highest support can still be identified by our algorithms (e.g. $\epsilon = 0.15$ in subfigure (b)).

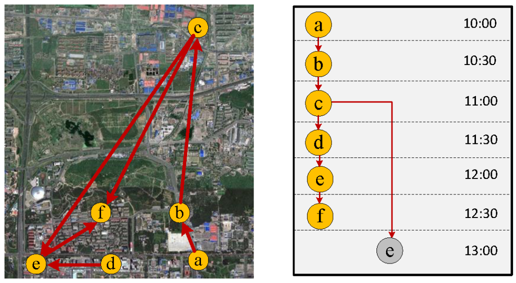
with highest support) subtrees are the ones that have strategic design drawbacks from the perspective of urban road network planning. For example, when $k = 7$, the top two frequent subtrees fall in the suburb of "Wangjing" and "Laoshan" respectively, shown in Figure 12. These subtrees indicate that both of the two suburbs are much more frequently overloaded with vehicles than other suburbs, and are in need of public transportation systems (e.g. subways) passing through them to reduce the need of commutations on ground. Such indications coincide with the future subway construction plan[4] of Beijing city: (i) New subway lines of Line 14 and Line 15 will be launched to travel through the suburb of "Wangjing" in year 2011 and 2013 respectively; (ii) Subway Line 10 (second construction stage) to be put into use by year 2012 will be centered at the suburb of "Laoshan".

These relationships between the official subway construction plans and our frequent subtrees validate the correctness of the `frequentSubtree` algorithm, and demonstrate the capability of our model in detecting design flaws in existing road networks.

[4] `http://en.wikipedia.org/wiki/Beijing_Subway`



(a) The most frequent subtree with support 12.5% covering the suburb of "Wangjing". Note that there are several loops appearing in this tree, which indicates circular causalities among certain regions.



(b) The second most frequent subtree with support 11.3% covering the suburb of "Laoshan".

Figure 12: Top two frequent substrees and suburbs they cover ("Wangjing" and "Laoshan") when $k = 7$. These two subtrees suggest that there are potential design flaws in the current road networks spanning the two suburbs. These results coincide with (and hence are validated by) future construction plan of Beijing subways.

6. CONCLUSIONS AND FUTURE WORK

In this paper we have studied the problem of detecting spatio-temporal outlier and their causal interactions from traffic data streams. We have proposed `STOTree`, an algorithm for discovering spatio-temporal outliers and causal relationships between them. While the worst-case time complexity of `STOTree` is quadratic (in the number of outliers in each time frame), empirical evidence strongly suggests that the complexity is closer to linear time.

We have also proposed a `frequentSubtree` algorithm which can be used to reveal recurrent anomalies in the road network. Based on the `STOTree` and `frequentSubtree` algorithm we were able to identify real and valid instances of anomalies in Beijing traffic data. This suggests that our approach has the potential of contributing to a new data driven approach towards road traffic analysis.

In future our plan is to apply and extend the use of our algorithms in the domain of internet traffics etc.

7. REFERENCES

[1] D. Brauckhoff, K. Salamatian, and M. May. Applying PCA for traffic anomaly detection: Problems and solutions. In *Proceedings of the 28th IEEE International Conference on Computer Communications (INFOCOM'09)*, pages 2866–2870, 2009.

[2] Y. Bu, L. Chen, A. Fu, and D. Liu. Efficient anomaly monitoring over moving object trajectory streams. In *Proceedings of the 15th SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'09)*, pages 159–168, 2009.

[3] H. Cao, D. Cheung, and N. Mamoulis. Discovering partial periodic patterns in discrete data sequences. *Advances in Knowledge Discovery and Data Mining (PAKDD 2004)*, pages 653–658, 2004.

[4] H. Cao, N. Mamoulis, and D. Cheung. Discovery of periodic patterns in spatiotemporal sequences. *Knowledge and Data Engineering, IEEE Transactions on*, 19(4):453–467.

[5] H. Cao, N. Mamoulis, and D. Cheung. Mining frequent spatio-temporal sequential patterns. In *Proceedings of the 5th IEEE International Conference on Data Mining (ICDM'05)*, pages 82–89, 2005.

[6] F. Chen, C.-T. Lu, and A. P. Boedihardjo. Gls-sod: a generalized local statistical approach for spatial outlier detection. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD'10)*, pages 1069–1078, 2010.

[7] L. Chen, M. Ozsu, and V. Oria. Robust and fast similarity search for moving object trajectories. In *Proceedings of the 24th ACM SIGMOD International Conference on Management of Data (SIGMOD'05)*, pages 491–502, 2005.

[8] M. Das and S. Parthasarathy. Anomaly detection and spatio-temporal analysis of global climate system. In *Proc. of the Third International Workshop on Knowledge Discovery from Sensor Data*, pages 142–150, 2009.

[9] M. Elfeky, W. Aref, and A. Elmagarmid. Periodicity detection in time series databases. *IEEE Transactions on Knowledge and Data Engineering*, pages 875–887, 2005.

[10] M. Elfeky, W. Aref, and A. Elmagarmid. WARP: time warping for periodicity detection. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pages 8–17, 2005.

[11] R. Estkowski. No Steiner point subdivision simplification is NP-Complete. In *Proceedings of the 10th Canadian Conference on Computational Geometry*, pages 11–20, 1998.

[12] S. Hirose, K. Yamanishi, T. Nakata, and R. Fujimaki. Network anomaly detection based on eigen equation compression. In *Proceedings of the 15th SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'09)*, pages 1185–1194, 2009.

[13] A. Lakhina, M. Crovella, and C. Diot. Diagnosing network-wide traffic anomalies. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM'04)*, pages 219–230, 2004.

[14] J. Lee, J. Han, and X. Li. Trajectory outlier detection: A partition-and-detect framework. In *Proceedings of the 24th International Conference on Data Engineering (ICDE'08)*, pages 140–149, 2008.

[15] J. Lee, J. Han, and K. Whang. Trajectory clustering: a partition-and-group framework. In *Proceedings of the 26th ACM SIGMOD International Conference on Management of Data (SIGMOD'07)*, pages 593–604, 2007.

[16] Z. Li, B. Ding, J. Han, R. Kays, and P. Nye. Mining periodic behaviors for moving objects. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD'10)*, pages 1099–1108, 2010.

[17] M. Lippi, M. Bertini, and P. Frasconi. Collective Traffic Forecasting. *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD 2010)*, pages 259–273, 2010.

[18] A. C. Lozano, H. Li, A. Niculescu-Mizil, Y. Liu, C. Perlich, J. Hosking, and N. Abe. Spatial-temporal causal modeling for climate change attribution. In *Proceedings of the 15th SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'09)*, pages 587–596, 2009.

[19] N. Pelekis, I. Kopanakis, C. Panagiotakis, and Y. Theodoridis. Unsupervised Trajectory Sampling. *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD 2010)*, pages 17–33, 2010.

[20] H. Ringberg, A. Soule, J. Rexford, and C. Diot. Sensitivity of PCA for traffic anomaly detection. In *Proceedings of the 2007 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS'07)*, pages 109–120, 2007.

[21] A. Rosenfeld. Connectivity in digital pictures. *Journal of the ACM (JACM)*, 17(1):160, 1970.

[22] P. Sun and S. Chawla. On local spatial outliers. In *Proceedings of the Fourth IEEE International Conference on Data Mining (ICDM'04)*, pages 209–216, 2004.

[23] E. Wu, W. Liu, and S. Chawla. Spatio-temporal outlier detection in precipitation data. *Knowledge Discovery from Sensor Data*, pages 115–133, 2010.

[24] J. Yuan, Y. Zheng, X. Xie, and G. Sun. Driving with knowledge from the physical world. In *Proceedings of the 17th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'11)*, 2011.

[25] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang. T-drive: Driving directions based on taxi trajectories. In *Proceedings of the 18th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL GIS 2010)*, 2010.

[26] Y. Zhang, Z. Ge, A. Greenberg, and M. Roughan. Network anomography. In *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, pages 30–42, 2005.