MANTRA: A Scalable Approach to Mining Temporally Anomalous Sub-trajectories

Prithu Banerjee
Dept. of Computer Science
University of British Columbia
Vancouver, Canada.
banerjee.prithu@gmail.com

Pranali Yawalkar
Dept. of CSE
IIT Madras
Chennai, India.
pranali.yawalkar@gmail.com

Sayan Ranu
Dept. of CSE
IIT Madras
Chennai, India.
sayan@cse.iitm.ac.in

ABSTRACT

In this paper, we study the problem of mining *temporally anomalous sub-trajectory* patterns from an input trajectory in a scalable manner. Given the prevailing road conditions, a sub-trajectory is temporally anomalous if its travel time deviates significantly from the expected time. Mining these patterns requires us to delve into the sub-trajectory space, which is not scalable for real-time analytics. To overcome this scalability challenge, we design a technique called *MANTRA*. We study the properties unique to anomalous sub-trajectories and utilize them in MANTRA to iteratively refine the search space into a disjoint set of sub-trajectory *islands*. The expensive enumeration of all possible sub-trajectories is performed only on the islands to compute the answer set of *maximal anomalous sub-trajectories*. Extensive experiments on both real and synthetic datasets establish MANTRA as more than 3 orders of magnitude faster than baseline techniques. Moreover, through trajectory classification and segmentation, we demonstrate that the proposed model conforms to human intuition.

1. INTRODUCTION

With the proliferation of cheap location tracking sensors, there is an abundance of data that are in the form of trajectories [10, 19]. Trajectory datasets have been mined for applications such as cellular network optimization [13], emergency detection [2], and taxi-route suggestions [19]. In this work, we study the problem of mining *maximal temporally anomalous sub-trajectories* in a scalable manner. Generally, an anomaly is defined as an observation O that deviates significantly from the rest of the data D. A trajectory is *temporally* anomalous if the time taken by it to cover a route, is significantly more, or less, than the typical time taken by the majority of the vehicles covering the same route.

To illustrate our problem, consider Fig. 1. It depicts the movement of 5 different trajectories through the same route starting at road segment 1 and ending at road segment 10 (x-axis). The y-axis denotes the time taken to cover the road segments. For example, the point X in the red trajectory T denotes that T has taken $0.75*60 = 45$ minutes to cover the first 3 road segments. Thus, the slope denotes the speed of the trajectory. It is clear that the dotted
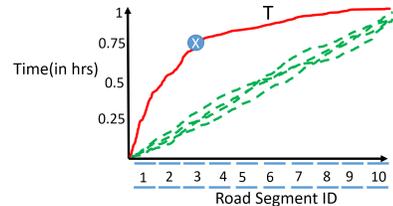

Figure 1: Illustration of the problem of mining maximal temporally anomalous sub-trajectories.

green trajectories follow a similar pattern and maintain a uniform speed. On the other hand, T moves much slower till segment 3, and thereon, it moves much faster than the rest of the population. Clearly, the driving pattern of T deviates from the rest. Mathematically, however, we do not reach the same conclusion. Specifically, T takes an hour to reach segment 10 just like all of the remaining trajectories. This erroneous conclusion can be corrected if we perform a sub-trajectory level analysis. More specifically, T's sub-trajectory from segment 1 to 3, denoted as $T[1:3]$, is temporally anomalous since T takes roughly 45 minutes to cover this distance as against the 15 minutes taken by the remaining trajectories. On the other hand, $T[4:10]$ is also temporally anomalous since it takes 15 minutes to cover this distance as opposed to 45 minutes taken by the remaining data. In other words, *a non-anomalous trajectory may contain temporally anomalous sub-trajectories.* Consequently, we need to delve into the sub-trajectory space.

A sub-trajectory level granularity also allows us to capture the semantics of the anomalies since a sub-trajectory could be anomalous due to *under-speeding* or *over-speeding*. Certainly, one can identify over-speeding or under-speeding by comparing vehicle speeds with the permissible speed limits in the traveled road segments. However, such an approach has three weaknesses. First, a vast majority of roads in developing countries, such as India, do not have any official speed limits. Second, even when speed limits are available, the "safe" driving speeds vary with traffic congestion levels, stoppage at traffic signals, weather conditions, road conditions, etc. Finally, beyond modeling, the number of sub-trajectories of a trajectory grows quadratically with the number of segments in the trajectory and poses a significant scalability challenge. The proposed solution addresses all of the above weaknesses.

A deeper analysis of the example depicted in Fig. 1 reveals that sub-trajectories $T[1:3]$ and $T[4:10]$ are not the only anomalous sub-trajectories. $T[1:2], T[4:9], T[5:9]$, and in general, all sub-trajectories of $T[1:3]$ and $T[4:10]$ are also anomalous. However, as they are all fully contained within $T[1:3]$ or $T[4:10]$, they do not provide any additional information over *maximal* anomalous

sub-trajectories $T[1:3]$ and $T[4:10]$. To remove this information redundancy, we only mine maximal anomalous sub-trajectories.

An obvious question arises at this juncture. *Why not simply identify contiguous stretches of road segments where a vehicle deviates from the rest?* Semantically, a maximal temporally anomalous sub-trajectory indicates the entire stretch of commute where a vehicle travels significantly faster or slower than the rest. Now, consider a maximal anomalous sub-trajectory S of a bus that covers a route in 15 minutes as opposed to 30 minutes taken by remaining population. However, within S, the bus was forced to stop at bus stops and traffic signals. Further, in some portions, bus was forced to follow remaining vehicles due to traffic congestion. Thus, although the bus traveled significantly faster than the rest in S, it was interspersed with non-anomalous road segments which do not necessarily imply safe intentions of the driver. This pattern is fairly common in urban conditions. Consequently, just identifying anomalous edges does not produce the entire stretch of commute where an anomalous pattern is observed and fails to capture the right semantics. So, higher order applications relying on anomalous patterns suffer.

Our algorithm provides powerful insights that can be leveraged in wide range of applications. The anomaly depicted by T in Fig. 1 is a typical scenario that leads to *Bus Bunching* [14]. In bus bunching, two or more buses that are scheduled to run on the same route at evenly spaced out time intervals, instead start to run together. One of the most common causes that leads to bus bunching is the revenue model where the driver's income is directly proportional to the number of passengers ferried. To maximize revenue, when no other buses are close by, the driver drives slowly to pick as many passengers as possible. By this time, if the next bus catches up, the driver accelerates in an attempt to stay ahead and not lose passengers to its competitor. By mining anomalous sub-trajectories, one can identify the drivers who indulge in such undesirable behavior. Further, Cab companies (ex. Uber), City buses and trucks install GPS in their vehicles. Our algorithm provides the power beyond merely rating the drivers. It enables to pinpoint where and how anomalous driving activities occurred, using the GPS traces. Furthermore, *car insurance* rates are calculated based on the risk profile of the driver. In such scheme, safe drivers gain through lower insurance premiums and companies gain through more accurate risk assessment. Currently assessment uses indirect factors such as education, gender, age etc. A more direct and accurate assessment using our algorithm, can be performed by mining the anomalous sub-trajectories in their historical records[1].

The above applications often need a real-time analysis. For example, users of cab services rate a driver as soon as their trips end. Users need to be informed about the anomalous portions before they submit their rating. In situations like bus-bunching, a real time monitoring system can be set up to warn the drivers involved in malicious practices and to improve passenger safety. To enable such real time analysis, the mining technique must scale.

To address the unsolved problem of mining maximal temporally anomalous sub-trajectories in a scalable manner, we develop a technique called *MANTRA (Maximal ANomalous sub-TRAjectories)*. Fig. 2 outlines the pipeline of MANTRA. The road network, in conjunction with the speed distributions, constitutes the background model. MANTRA derives its power by decomposing a trajectory into a set of *anomalous islands*. Anomalous islands are partitions of a trajectory such that all maximal anomalous sub-trajectories are contained in the set of islands. At the same time, no maximal anomalous sub-trajectory spans across two islands. Based on these properties, MANTRA performs a *bi-directional sliding win-*
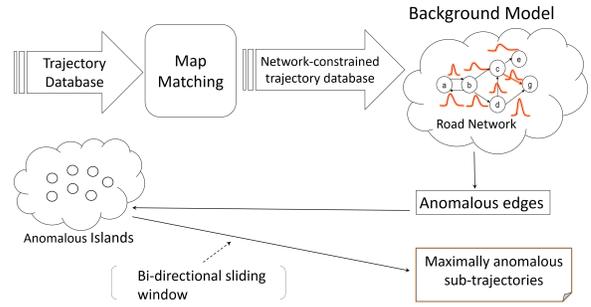
Figure 2: Pipeline of MANTRA.

dow search to mine maximal anomalous sub-trajectories from each island. To summarize, our main contributions are the following:

● We formulate the problem of mining *maximal temporally anomalous sub-trajectory* in trajectory dataset. We empirically validate, why identifying stretches of contiguous anomalous edges alone does not suffice from application context.

● To scale the problem for real-time analysis on trajectory, we design a technique called *MANTRA*. MANTRA avoids the expensive sub-trajectory enumeration step on the majority of the trajectory by identifying a small set of *islands*.

● Extensive experiments on real datasets show MANTRA as more than 3 orders of magnitude faster than baselines, effective in intuitive modeling of anomalous driving and up to 87% better in trajectory segmentation than using only anomalous edges.

2. PROBLEM FORMULATION

First, we define the concepts central to our paper.

DEFINITION 1. ROAD NETWORK. *A road network is a directed graph $G(V,E)$. V is the set of nodes representing intersections and terminal points, and E is the set of edges $e = (v_i, v_j)$, connecting $v_i, v_j \in V$, depicting road segments. The position of a node is characterized by its latitude and longitude.*

Trajectories are received in the form of GPS traces, that are temporally ordered sequences of spatio-temporal points. A spatio-temporal point $s = (v, t)$ is a tuple containing a spatial location v and a timestamp t encoding the time of the day at which v is traversed. These GPS-traces are *map-matched* to the road network G to construct *network-constrained* trajectories.

DEFINITION 2. NETWORK-CONSTRAINED TRAJECTORY. *T is constrained in a road network $G(V, E)$, if $\forall T.s_i$, $T.s_i.v \in V$, and $\forall T.s_i$, $T.s_{i+1}$, $(T.s_i.v, T.s_{i+1}.v) \in E$.*

In simple words, a network-constrained trajectory is a connected path in the road network. In the network-constrained form, we represent a trajectory $T = [st, (e_1, t_{e_1}), \cdots, (e_l, t_{e_l})]$ as an ordered sequence of edges and their corresponding travel times. $st = T.s_1.t$ denotes the time at which T started traveling. We use $T.e_i$ to denote the i_{th} edge visited by trajectory T and $T.t_{e_i}$ is the time taken to travel edge e_i. $e \in T$ denotes that T has traveled through e. $|T|$ denotes the size of T in terms of the number of edges traversed. A *sub-trajectory* $T[m:n]$ of a trajectory T is the sequence of edges from $T.e_m$ to $T.e_n$. We denote the sub-trajectory relationship using $S \subseteq T$, where $S = T[m:n]$ and $1 \leq m \leq |T|$, $m \leq n \leq |T|$. A *super-trajectory* is defined analogously.

Generally, a sub-trajectory S is anomalous if its commute time deviates significantly from the rest of the data $\mathbb{D}$. $\mathbb{D}$ characterize the prevailing traffic conditions. We adopt the standard *z-score*

based anomaly model [8]. Let the distribution of travel times in the route covered by S be $N(\mu_S, \sigma_S^2)$, where μ_S is the mean and σ_S^2 is the variance. To quantify S as anomalous, we compare how many variances (or standard deviations) away the travel time of S is from μ_S. Mathematically, for a sub-trajectory S, we construct the *background set* $BS(S) = \{T \in \mathbb{D} | S \subseteq T\}$. More simply, the background set of S contains all trajectories in $\mathbb{D}$ that contain S as a sub-trajectory. The variance of travel times in these edges is

$$\sigma_S^2 = \frac{1}{|BS(S)|} \sum_{\forall T \in BS(S)} (\mu_S - time_S(T))^2 \qquad (1)$$

$$\text{where } time_S(T) = \sum_{\forall e \in S} T.e_t \qquad (2)$$

$$\mu_S = \frac{1}{|BS(S)|} \sum_{\forall T \in BS(S)} time_S(T) \qquad (3)$$

For trajectory $T \in \mathbb{D}$, sub-trajectory $S \subseteq T$ is *anomalous* if

$$\frac{(\mu_S - time_S(S))^2}{\sigma_S^2} > \theta \qquad (4)$$

where θ is a user provided threshold. The interpretation of θ can be obtained from the z-score table [8]. For example, the probability of being more than 4 variance away from the mean, i.e. $p((\mu_S - time_S(S))^2 \geq 4\sigma_S^2) \approx 0.05$. Based on whether a 0.05 chance is low enough to be termed as anomalous, one can set θ.

DEFINITION 3. MAXIMAL ANOMALOUS SUB-TRAJECTORY. *A sub-trajectory S is* maximal anomalous sub-trajectory (MAS) *if $\nexists S' \subseteq T$, $S' \supset S$ and S' is anomalous.*

PROBLEM 1. *Let $\mathbb{D}$ be the reference dataset of trajectories. For each incoming trajectory, identify all of its maximal temporally anomalous sub-trajectories under a user-provided threshold θ with respect to $\mathbb{D}$.*

3. THE NAÏVE APPROACH

Given a trajectory T, the naïve approach is to enumerate all possible sub-trajectories of T, check if they are anomalous, and then filter out those anomalous sub-trajectories that are not maximal anomalous. This naïve approach however suffers from multiple issues. First, a trajectory T over n edges produces $O(n^2)$ sub-trajectories. $O(n^2)$ is not scalable in real-time settings. Second, computing the deviation from mean itself is often not possible due to data sparsity. Consider the database of trajectories in Fig. 3(a). Let us focus on T_3 in Fig. 3(a). For the sub-trajectory $S = \{b, d, c, e\}$ of T_3, $BS(S) = \emptyset$. As a result, the mean and the variance in this background set cannot even be computed. This is a common problem when sub-trajectory sizes are large. We establish this observation empirically. We pick random sub-trajectories from the *T-drive* [19] trajectory dataset and compute the average size of their background sets. As it can be seen in Fig. 3(b), the size of the background set plummets with the size of the sub-trajectory. This analysis gives us one key insight; if the sub-trajectory is small, then the z-score can be computed reliably. Else, we need an alternative.

4. MANTRA

Sec. 3 outlines the two key challenges of the proposed problem: data sparsity and scalability. First, we address the modeling issue.
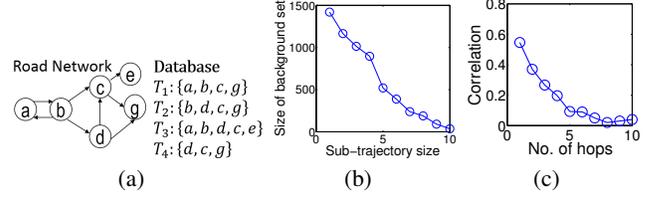

Figure 3: **(a) A road network and a database of network-constrained trajectories. For simplicity, we ignore the timestamps. (b) Illustration of data sparsity with sub-trajectory size. (c) Correlation among edges against the distance between them.**

4.1 Managing data sparsity

Revisiting Fig. 3(a), we notice that although no trajectory in $\mathbb{D}$ contains $S = \{b, d, c, e\}$, all edges in S have been visited by at least one trajectory. In this situation, a human being would estimate the expected time to cover S by aggregating the individual pieces of information on each edge. As shown in Fig. 3(b), there is enough data for individual edges, i.e., sub-trajectories of length 1, and data sparsity is not an issue in this 1-dimensional space. MANTRA builds on this same intuition to compute the *expected time*. Let $\mathbb{D}$ be the set of historical trajectories that were travelled on road network $G(V, E)$. For each edge $e \in E$, we fit a normal distribution $time(e) = \mathcal{N}(\mu_e, \sigma_e^2)$ to approximate the distribution of travel times in e. Normal distributions have been shown to characterize traffic speed distributions well [9]. Thus, for a road network $G = (V, E)$, we learn a set $\mathbb{N} = \bigcup \mathcal{N}(\mu_e, \sigma_e^2), \forall e \in E$. $\mathbb{N}$ encodes the ongoing traffic conditions.

We model distribution of travel times in sub-trajectory S as a multivariate distribution of its constituent edges. Thus, $time(S) \approx \sum_{\forall e \in S} time(S.e) = \mathcal{N}(\mu_S, \sigma_S^2)$, where $\mu_S = \sum_{\forall e \in S} \mu_e$ and

$$\sigma_S^2 = \sum_{\forall e \in S} \sigma_e^2 + 2 \sum_{\forall \{e, e'\} \in S} cov(e, e') \qquad (5)$$

$cov(e, e')$ is the covariance and it captures the dependence between travel times of the two edges e and e'.

In the above formulation, the term $\sum_{\forall \{e, e'\} \in S} cov(e, e')$ requires storing the covariance matrix on the entire edge set of the road network. Storing the entire covariance matrix is not feasible due to both storage costs and the computation cost of updating them at the arrival of every new trajectory. However if two edges are far apart from each other, then they are likely to be independent in their traffic flow. We show this empirically in Fig. 3(c). It shows the results and brings out two key properties.
1. Dependency exists between edges within 5 hops.
2. $\forall e, e' \in E, cov(e, e') \geq 0$.

The correlation is highest between edges that are immediate neighbors. As we move away from the edge, the correlation drops, and beyond 5, it is close to 0. Furthermore, we do not have negative correlation among edges. A negative correlation between edges x and y would mean that if a car takes more than expected time in x, then it would take less than expected time in y. Such a dependence is rare. The more common scenario is where a congestion spreads to neighboring edges or eases out which result in positive covariance. To gain more confidence on the absence of negative correlation, we remove the hop-based constraint and check correlation among 10000 randomly picked edge pairs. We found a correlation close to 0 in this study as well.

Armed with this observation, we integrate them in the MANTRA

framework for more efficient and accurate anomaly detection. We utilize the first property in condensing the covariance matrix. Specifically, for each edge we store the covariance matrix with all edges in its 5-hop neighborhood. Whereas non-existence of negative covariance is utilized later to prune the search space.

4.2 Fine-tuning the z-score based anomaly model

We showed earlier that the travel time in a sub-trajectory S can be estimated by aggregating the distributions in its constituent edges. Applying z-score on this expected time of S, however, may produce the following counter-intuitive and impractical result.

THEOREM 1. *S can be anomalous even if the travel times in all of its constituent edges are non-anomalous.*

PROOF: Consider $S = (e_1, t_{e_1}), (e_2, t_{e_2})$. Let $(S.t_{e_1} - \mu_{e_1})^2 = \theta\sigma_{e_1}^2$ and $(S.t_{e_2} - \mu_{e_2})^2 = \theta\sigma_{e_2}^2$ and $cov(e_1, e_2) = 0$. Thus, both edges are non-anomalous. However, S is anomalous since

$$\frac{(S.t_{e_1} + S.t_{e_2} - \mu_S)^2}{\sigma_S^2} = \frac{\theta(\sigma_{e_1} + \sigma_{e_2})^2}{\sigma_{e_1}^2 + \sigma_{e_2}^2} > \theta \qquad \square$$

Let us now analyze the implications of the theorem in a real life scenario. Essentially, $\sqrt{\theta}\sigma_e$ is the maximum time by which one can deviate from μ_e in each road segment e, and still be non-anomalous in that edge. This is analogous to driving within the permissible speed limits. Now, Theorem 1 shows that a driver who drove non-anomalously in all segments could still end up being classified as anomalous in sub-trajectory S. Clearly, such an anomaly model goes against the human intuition. We therefore need a model where for a sub-trajectory to be anomalous, it must contain at least one edge with an anomalous travel time. Towards that, we define the deviation of S by aggregating deviations in its constituent edges.

$$dist(S) = \sum_{\forall e \in S} \mathcal{I}(e)d_e^2 \qquad (6)$$

where $d_e = |\mu_e - S.t_e|$ is the deviation from the expected time at edge e and $\mathcal{I}(e)$ is an indicator function detecting under-speeding or over-speeding.

$$\mathcal{I}(e) = \begin{cases} 1 & \text{if } S.t_e >= \mu_e \\ -1 & \text{if } S.t_e < \mu_e \end{cases} \qquad (7)$$

S is *anomalous* if

$$\frac{|dist(S)|}{\sigma_S^2} > \theta \qquad (8)$$

We call an anomalous sub-trajectory S as *positively* anomalous if $dist(S)$ is positive, and *negatively* anomalous otherwise.

THEOREM 2. *Based on the anomaly definition of Eq. 8, a sub-trajectory S cannot be anomalous unless it has at least one anomalous travel time in an edge.*

PROOF: Consider the case where the travel times in each edge deviates the most from their means without being anomalous and $\forall e, \mathcal{I}(e) = 1$. Therefore, $d_e^2 = \theta\sigma_e^2$. $dist(S) = \sum_{\forall e \in S} \theta\sigma_e^2 = \theta\sigma_S^2$. S is non-anomalous since $\frac{dist(S)}{\sigma_S^2 + \Delta} \leq \theta$ where $\Delta \geq 0$ is the contribution from covariance. $\qquad \square$

EXAMPLE 1. *Fig. 4 shows a sample trajectory T and the MASs contained in it. For each edge, Fig. 4 also shows the $I(e)d_e^2$ component to the $dist(T)$ function (Eq. 8). Assume $\forall e, e', \sigma_e^2 = 1, cov(e, e') = 0, \theta = 1$. Thus, a sub-trajectory $S \subseteq T$ is anomalous if the average of $\mathcal{I}(e)d_e^2$ in its edges is larger than 1 or less than -1.*
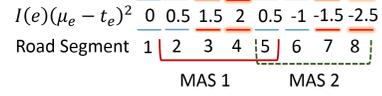

| $I(e)(\mu_e - t_e)^2$ | 0 | 0.5 | 1.5 | 2 | 0.5 | -1 | -1.5 | -2.5 |
| Road Segment | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

MAS 1   MAS 2

Figure 4: The MASs in the shown example. Red edges denote the anomalous edges ($|I(e)d_e^2| > 1$).

Algorithm 1 BSW(T)

```
1:  M ← ∅
2:  i ← 1
3:  rightSoFar ← i
4:  while i < T.size() do
5:      if rightSoFar == T.size() then
6:          break
7:      j ← T.size()
8:      while j > rightSoFar do
9:          if isAnomalous(T[i : j]) then
10:             add T[i : j] in M
11:             rightSoFar ← j
12:             break
13:         j ← j − 1
14:     i ← i + 1
15: return M
```

4.3 Mining maximal anomalous sub-trajectories

The naïve algorithm described in Sec. 3 is agnostic to mining only maximal anomalous trajectories. More specifically, enumerating all possible sub-trajectories will lead us to candidates that are anomalous, but not maximal anomalous. To illustrate, consider a trajectory T where the travel times in all of its edges are anomalous. Here, if we first check for anomaly on T itself, then we will avoid the entire $O(|T|^2)$ cost of enumerating its sub-trajectories since none of them can be maximal anomalous. To extend this same intuition, it is beneficial to evaluate larger sub-trajectory candidates first since a sub-trajectory $S \subseteq T$ being anomalous removes all of its $O(|S|^2)$ sub-trajectories from the $O(|T|^2)$ candidate space.

4.3.1 Bi-directional sliding window

We develop a *bi-directional sliding window* technique that builds on the intuition that it may be beneficial to evaluate larger sub-trajectory candidates first. Alg. 1 outlines the pseudocode. Given a trajectory T, we maintain two pointers pointing to the two extreme ends of the trajectory (lines 2 and 7). These pointers indicate the boundaries of the sub-trajectory that is being evaluated for anomalous behavior. Initially, this sub-trajectory is T itself. Iteratively, we refine to smaller trajectories.

Revisiting Fig. 4, initially, the two pointers i and j point to 1 and 8 respectively. The sub-trajectory within this boundary is the entire trajectory itself, and we check if it is anomalous. Since, it is not, keeping i fixed at 1, we decrement j iteratively till we either find an anomalous sub-trajectory or j hits i (lines 8-13). At $i = 1$, no anomalous trajectory is found. Next, i is incremented by 1 (line 14) to $i = 2$ and j is reset to 8. Repeating the same procedure, an anomalous sub-trajectory is found at $j = 5$. It is guaranteed that this trajectory is a MAS since all its super-trajectories have already been evaluated. In addition, for any other MAS that has not yet been discovered, it must end beyond $j = 5$ (line 11). Thus, we restart the search procedure by resetting $j = 8$ and incrementing $i = 3$. However, we need to search for j only in the range [6:8]. Following this procedure, we end when either $i = 7$ (line 4) or a MAS is found that ends at edge segment 8 (lines 5-6). The second case occurs in Fig. 4 when MAS 2 is found between segments 5 and 8. To summarize, Alg. 1 guarantees the following.

Figure 5: A trajectory where the bi-directional sliding window algorithm is not efficient. Red edges denote anomalous ones.

Figure 6: The islands contained in the trajectory in Fig. 5

THEOREM 3. *Alg. 1 does not evaluate any anomalous sub-trajectory that is non-maximal.*

From Theorem 3, it is clear that the larger the size of the MASs in a trajectory, the less is the computation cost of Alg. 1. However, the sliding window technique is not always efficient. Consider the scenario in Fig 5. Here, the bi-directional sliding window technique performs almost as worse as naïve since the MASs are all small, and consequently, does not prune out too many sub-trajectory candidates. Furthermore, since the MASs are evenly spaced out, at most i and j, $T[i:j]$ has at least one anomalous edge within it and consequently cannot be pruned out.

A closer analysis of Fig. 5 indicates that the MASs are small in size and the number of anomalous edges is less. So it is more efficient to start search procedure from around the anomalous edges itself, and expand the search space by expanding to their neighboring edges till the MASs are found. Such an approach, however, is diametrically opposite to the philosophy of the sliding window technique. Not only does this procedure generate non-maximal anomalous sub-trajectories, since two MASs can overlap, as in MAS 2 and MAS 3 of Fig. 5, the expansion procedure itself is complex. More critically, this approach suffers when the MASs are large.

The above discussion reveals that both the sliding window technique and the neighborhood expansion approach have their weaknesses. While they work well in certain situations, their efficiencies can deteriorate based on the distribution of the anomalous edges in a trajectory. We need a technique that is more robust to all distributions. Towards that goal, MANTRA adopts the unique approach of partitioning a trajectory into *islands* such that by performing sliding window on the islands only, all MASs can be mined. In Fig. 5, there are four islands, which are shown in Fig. 6. The islands are mined by performing a neighborhood expansion strategy, which we formalize in the next section. An island is likely to contain MASs that span almost the entire length of the island, and therefore, sliding window in an island is extremely efficient.

4.3.2 Mining islands

An *island* is essentially a sub-trajectory of trajectory T with special properties. Specifically, our goal is to identify a set of disjoint sub-trajectories, or islands $\mathbb{I}$, that satisfies the following property in terms of maximal anomalous sub-trajectories $\mathbb{M}$ in T.

$$\forall S \in \mathbb{M}, \exists I \in \mathbb{I} \text{ such that } S \subseteq I. \qquad (9)$$

In other words, it is enough to perform sliding window only on the islands in $\mathbb{I}$. It is easy to see that all anomalous edges must be contained in some island. The more non-trivial part is to identify the non-anomalous edges that could potentially be part of some MAS. A non-anomalous edge becomes part of an anomalous sub-trajectory when the excess anomaly from an anomalous edge spills over to its neighbors. MANTRA builds on this intuition.

To formalize the approach of MANTRA, we first introduce few terminologies. Two sub-trajectories $T_1[m_1 : n_1]$ and $T_2[m_2 : n_2]$ of T are said to be *meeting* if either $m_2 = n_1 + 1$ or $m_1 = n_2 + 1$. Two sub-trajectories $T_1[m_1 : n_1]$ and $T_2[m_2 : n_2]$ are said to be *overlapping* if either $m_2 < n_1 < n_2$ or $m_1 < n_2 < n_1$. Finally, *synergy* of k ($k \geq 2$) sub-trajectories $T_1[m_1 : n_1], T_2[m_2 :$

$n_2], \cdots, T_k[m_k : n_k]$, denoted using the notation $\psi(T_1, \cdots, T_k)$, is the sub-trajectory $T_{1k}[min(m_1, \cdots, m_k) : max(n_1, \cdots, n_k)]$. More simply, synergy is the join of the k sub-trajectories.

EXAMPLE 2. *Let T be the trajectory in Fig. 5. In T, MAS 2 and MAS 3 are overlapping, but not meeting. MAS 4 is meeting with $T[8 : 10]$. $\psi(MAS\ 2, MAS\ 3) = T[6 : 8]$. The $\psi(MAS\ 1, MAS\ 2, MAS\ 3) = T[2 : 8]$. MAS 1 is neither meeting nor overlapping with MAS 2.*

Alg. 2 outlines the pseudocode of MANTRA. MANTRA first extracts the contiguous stretches of anomalous edges in a given trajectory T (line 2). For example, in Fig. 4, these stretches are $S_1 = T[3 : 4]$ and $S_2 = T[7 : 8]$. We call these the *seeds* since the excess anomalies in these seeds spill over to neighboring edges. To understand the impact of these seeds, we need to bound the extent of this spill-over effect. For that purpose, we introduce the definitions of *left boundary* and *right boundary*.

DEFINITION 4. LEFT BOUNDARY. *The left boundary of a sub-trajectory $S = T[i : j] \subseteq T$, denoted as $LB(S)$, is the sub-trajectory $T[l : j] \subseteq T$ such that $\forall v, l \leq v \leq i, T[v : j]$ is anomalous and either $T[l - 1, j]$ is not anomalous or $l = 1$.*

The *right boundary* of a sub-trajectory S, denoted as $RB(S)$, is defined analogously. $LB(S)$ essentially bounds the maximal spill-over effect on the left side and $RB(S)$ bounds it on the right side.

EXAMPLE 3. *At $\theta = 1$, the two seeds in Fig. 4 are $S_1 = T[3 : 4]$ and $S_2 = T[7 : 8]$. $LB(S_1) = T[2 : 4]$ and $RB(S_1) = T[3 : 5]$. $LB(S_2) = T[5 : 8]$ and $RB(S_2) = T[7 : 8]$.*

Since LB and RB define the extent of the anomaly spread on either side of a seed S, $\psi(LB(S), RB(S))$ bounds the entire range within which the impact of S is limited. Therefore, after identifying the seeds (line 2), MANTRA computes the LB and RB of each one of them (lines 1-3 in Alg. 3). Now, if the *impact region* of each seed, which is the synergy of its LB and RB, is unaffected by the impact regions of the remaining seeds, then we can essentially guarantee that any MAS must be contained within the impact regions of the seeds. However, the impact regions of two seeds S_1 and S_2 can interact giving rise to various unique situations:

1. $\psi(LB(S_1), RB(S_1))$ and $\psi(LB(S_2), RB(S_2))$ overlap.
2. $\psi(LB(S_1), RB(S_1))$ meets $\psi(LB(S_2), RB(S_2))$.
3. $\psi(LB(S_1), RB(S_1))$ and $\psi(LB(S_2), RB(S_2))$ are separated by non-anomalous edges.
4. $\psi(LB(S_1), RB(S_1)) \subseteq \psi(LB(S_2), RB(S_2))$.
5. $\psi(LB(S_1), RB(S_1)) \supseteq \psi(LB(S_2), RB(S_2))$.

Case 4 and Case 5 are easiest to manage. Specifically, since the impact region of one seed is contained within the other, the seed with the smaller impact region can be discarded (lines 10-14 in Alg. 2). An example of this scenario is shown in Fig. 7 where the impact region of seed ST_2 is contained with impact region of ST_3. The interactions between impact regions in cases 1-3 are more complicated. We first analyze case 1 where two impact regions, which are potentially anomalous sub-trajectories of a trajectory, overlap.

THEOREM 4. *Synergy S of two overlapping anomalous sub-trajectories S_1 and S_2 may or may not be anomalous.*

PROOF: Let S_1 and S_2 be positively anomalous and all edges have 0 covariance between them. We denote overlapped portion as S_{12} and non-overlapping portions as S_{11} in S_1 and S_{22} in S_2. Clearly, S_{11}, S_{12} and S_{22} are meeting. Assume overlapping portion S_{12} is anomalous, whereas non-overlapping portions S_{11} and S_{22} are not.

Case 1: We consider the case where for the non-overlapping portions $\frac{dist(S_{11})}{\sigma_{S_{11}}^2} = \frac{dist(S_{22})}{\sigma_{S_{22}}^2} = \theta$. On the other hand, $\frac{dist(S_{12})}{\sigma_{S_{12}}^2} = \theta + c_{12}$, where $c_{12} > 0$. Now, since S is the synergy of three meeting sub-trajectories S_{11}, S_{12} and S_{22}, for S,

$$\frac{dist(S)}{\sigma_S^2} = \frac{dist(S_{11}) + dist(S_{12}) + dist(S_{22})}{\sigma_S^2}$$
$$= \frac{\theta(\sigma_{S_{11}}^2 + \sigma_{S_{12}}^2 + \sigma_{S_{22}}^2) + \sigma_{S_{12}}^2 c_{12}}{\sigma_S^2} > \theta$$

Case 2: Since S_1 and S_2 are anomalous, let $\frac{dist(S_1)}{\sigma_{S_1}^2} = \theta + \epsilon$ and $\frac{dist(S_2)}{\sigma_{S_2}^2} = \theta + \epsilon$, where $\epsilon \approx 0$, but positive. Further, let $dist(S_{11}) = dist(S_{22}) = 0$. Therefore, $dist(S_{12}) = dist(S_1) = (\theta + \epsilon)\sigma_{S_1}^2 = dist(S_2) = (\theta + \epsilon)\sigma_{S_2}^2$. Thus, for the synergy,

$$\frac{dist(S)}{\sigma_S^2} = \frac{dist(S_{11}) + dist(S_{12}) + dist(S_{22})}{\sigma_S^2}$$
$$= \frac{(\theta + \epsilon)\sigma_{S_1}^2}{\sigma_S^2} \leq \theta \text{ whenever } \epsilon\sigma_{S_1}^2 \leq \theta\sigma_{S_2}^2 \ \square$$

Theorem 4 essentially shows that if the impact regions of two seeds S_1 and S_2 overlap, then there could be a MAS that spanning $\psi(\psi(LB(S_1), RB(S_1)), \psi(LB(S_2), RB(S_2)))$.

Next, we analyze case 2 where two impact regions meet.

THEOREM 5. *Synergy of non-anomalous sub-trajectories S_1, S_2, which are meeting, is also non-anomalous.*

PROOF: From Eq. 8, $dist(S_1) \leq \theta\sigma_{S_1}^2$ and $dist(S_2) \leq \theta\sigma_{S_2}^2$. Thus, $\frac{dist(\psi(S_1,S_2))}{\sigma_{S_1}^2 + \sigma_{S_2}^2 + \Delta} \leq \theta$, where $\Delta = cov(S_1, S_2) \geq 0$. $\square$

It is easy to see that the above theorems also hold for negatively anomalous sub-trajectories. If the sub-trajectories are a mixture of positively and negatively anomalous sub-trajectories then no definitive conclusions can be drawn. Thus, like in the case of overlap, interactions are possible between two meeting impact regions, which can result in a MAS spanning both these impact regions. Finally, we analyze the third case where two impact regions are separated by non-anomalous edges.

THEOREM 6. *If the number of separating non-anomalous edges between two impact regions I_1, I_2 is more than one, then the synergy can never be anomalous.*

PROOF: Assume a case where there are exactly two non-anomalous edges e_1 and e_2 between I_1 and I_2. Without loss of generality we assume e_1 is adjacent to I_1 and e_2 and e_2 is adjacent to e_1 and I_2. From the construction of our impact regions, no impact region can be expanded by one non-anomalous edge and remain anomalous. Thus, $\psi(I_1, e_1)$ and $\psi(I_2, e_2)$ are both non-anomalous. These two synergies are again two meeting non-anomalous sub-trajectories and following Theorem 5, they can never be anomalous. $\square$

THEOREM 7. *Synergy of two impact regions separated by at most one non-anomalous edge can be anomalous.*

PROOF: Let the two impact regions be I_1 and I_2 and separated by a non-anomalous edge e and $cov(I_1, I_2) = 0$. From the construction of our impact regions, we know that they cannot be expanded

---

Algorithm 2 MANTRA(T)

1: $ST \leftarrow \emptyset$
2: $CG \leftarrow$ set of all contiguous sequences of anomalous edges
3: **while** $\forall S \in CG$ **do**
4:     addSeed(S, ST)
5: **repeat**
6:     **for** $\forall T_i = T[m_i : n_i] \in ST$ **do**
7:         $SYN_i \leftarrow \psi(T_i.LB, T_i.RB)$
8:         **for** $\forall T_j = T[m_j : n_j] \in ST$, such that $m_j > m_i$ **do**
9:             $SYN_j \leftarrow \psi(T_j.LB, T_j.RB)$
10:             **if** $SYN_i \subseteq SYN_j$ **then**
11:                 $ST \leftarrow ST\backslash\{T_i\}$
12:                 break
13:             **else if** $SYN_j \subseteq SYN_i$ **then**
14:                 $ST \leftarrow ST\backslash\{T_j\}$
15:             **else if** SYN_i and SYN_j are overlapping, meeting or separated by one edge **then**
16:                 $ST \leftarrow ST\backslash\{T_i, T_j\}$
17:                 $SYN_{ij} \leftarrow \psi(SYN_1, SYN_2)$
18:                 addSeed(SYN_{ij}, ST)
19:                 break;
20: **until** convergence of ST
21: $\mathbb{M} \leftarrow \emptyset$
22: **for** $S \in ST$ **do**
23:     $\mathbb{M} \leftarrow \mathbb{M} \cup \textbf{BSW}(\psi(S.LB, S.RB))$
24: **return** $\mathbb{M}$

---

further by an edge while keeping it anomalous. Thus, the maximum possible deviation is bounded by $\frac{dist(I_1) + dist(e)}{\sigma_{I_1}^2 + \sigma_e^2} = \theta$ and $\frac{dist(I_2) + dist(e)}{\sigma_{I_2}^2 + \sigma_e^2} = \theta$. Therefore,

$$dist(I_1) = \theta(\sigma_{I_1}^2 + \sigma_e^2) - dist(e) \tag{10}$$
$$dist(I_2) = \theta(\sigma_{I_2}^2 + \sigma_e^2) - dist(e) \tag{11}$$

Since e is non-anomalous,

$$dist(e) \leq \theta\sigma_e^2 \tag{12}$$

Clearly, I_1, e, I_2 are meeting sub-trajectories. Thus, for their synergy sub-trajectory S, we have

$$\frac{dist(S)}{\sigma_S^2} = \frac{dist(I_1) + dist(e) + dist(I_2)}{\sigma_S^2}$$
$$= \frac{\theta(\sigma_{I_1}^2 + \sigma_e^2) - dist(e) + dist(e) + \theta(\sigma_{I_2}^2 + \sigma_e^2) - dist(e)}{\sigma_S^2}$$
$$= \theta + \frac{\theta\sigma_e^2 - dist(e)}{\sigma_e^2} > \theta \text{ by applying the inequality in Eq. 12}$$

Combining Theorems 7 and 6, we can guarantee that two impact regions can interact only if they are separated by at most one non-anomalous edge. Thus, based on Theorems 4-7, interactions between impact regions can happen only if they are overlapping, meeting, or separated by one non-anomalous edges. With this deeper understanding of the behavior of anomalous sub-trajectories, we are now ready to formalize the MANTRA algorithm.

MANTRA starts by identifying the contiguous stretches of positive anomalous edges or negative anomalous edges (line 2). These form the initial seeds and their left and right boundaries are computed and stored. The seeds are stored sorted based on their starting indices. For the scenario shown in Fig. 7, the corresponding stored items are shown in Table. 1. Next, MANTRA iterates over each seed

---

Algorithm 3 addSeed(S, ST)

1: $ST \leftarrow ST \cup \{S\}$
2: $S.LB \leftarrow \textbf{LB}(S)$
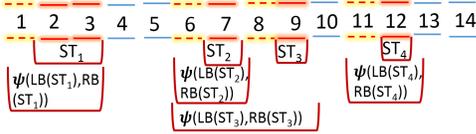3: $S.RB \leftarrow \textbf{RB}(S)$

Figure 7: Illustration of seeds, left boundary, right boundary, and synergy. Red glowing edges denote anomalous edges. Yellow glowing dashed edges denote non-anomalous edges that are in the left or right boundary of an anomalous edge. Refer to Table. 1 for the corresponding stored items in the seed set.

ST	LB	RB
$ST1 = T[2:3]$	$T[1:3]$	$T[2:3]$
$ST2 = T[7:7]$	$T[6:7]$	$T[7:7]$
$ST3 = T[9:9]$	$T[6:9]$	$T[9:9]$
$ST4 = T[12:12]$	$T[11:12]$	$T[12:12]$

Table 1: Seed set and their boundaries for the example in Fig.7

and matches their impact regions (lines 5-20). If one impact region is found to be a subset of the other, then the smaller seed along with its boundaries are discarded from the seed set (lines 10-14). Otherwise, if the impact regions of two seeds S_1 and S_2 are found to be interacting, then their synergies are computed. This synergy now forms a new seed. Furthermore, S_1 and S_2 are discarded from the seed set (lines 15-19). Once all impact regions have been analyzed and the new seeds have been identified, MANTRA re-starts the same analysis on the new larger seeds by computing their impact regions. The process continues till the impact regions of all existing seeds are found to be non-interacting. These non-interacting impact regions are the islands, and thus, sliding window is performed on these islands to compute the MASs.

EXAMPLE 4. *The initial items in the seed set for Fig. 7 is shown in Table 1. From this set, the seed ST_2 is discarded since its impact region is a subset of the impact region of ST_3. ST_1 remains in the seed set since its impact region is non-interacting with other impact regions. However, a new seed is formed in the second iteration by taking the synergy of the impact regions of ST_3 and ST_4 since they are separated by only one anomalous edge. After the second iteration, there are two seeds: ST_1 and sub-trajectory $[6:12]$.*

THEOREM 8. PROOF OF CORRECTNESS: *Every MAS is contained in the set of islands.*

PROOF: Our method of constructing islands ensures that every anomalous edge is part of some island. Hence there is no anomalous edge left out outside of our island boundary. Moreover, in MANTRA, the seed expansion is stopped only when there are at least two non-anomalous edges separating the impact regions of any pair of seeds. This guarantees non-interaction between the final impact regions, which are our islands (Theorem 5). Thus, no MAS can span across islands. □

5. EXPERIMENTS

In this section, we empirically demonstrate the efficiency and efficacy of MANTRA. Our algorithms are implemented in Java JDK 1.7.0 and evaluated on a PC with 12GB memory and Intel i5 2.60GHz quad core processor running Ubuntu 13.04.

Datasets: We use GPS-traces from two different sources: T-Drive [19] and GeoLife [21]. Both traces are from Beijing. T-Drive contains 136,759 trajectories and GeoLife contains 18,670 trajectories. While GeoLife is smaller, it is richer in terms of trajectory diversity containing traces from buses, cars, and walks. Road network of Beijing, which contains 623,975 nodes and 672,284 edges, is extracted from OpenStreetMap. The distribution of the trajectory sizes is shown in Fig. 8(a).

Experimental Setup: We benchmark scalability of MANTRA against the Naïve approach described in Sec. 3, and Sliding Window (Alg. 1) on the entire trajectory instead of just the islands.

Unless specifically mentioned, we set $\theta = 4$. In a normal distribution $\mathcal{N}(\mu, \sigma^2)$, the probability of being more that 4 variance away, i.e. $p((x - \mu)^2 \geq 4\sigma^2) \approx 0.05$, which is commonly considered as statistically significant.

5.1 Scalability

First, we evaluate the growth rate of the running time against trajectory size (number of edges). Fig. 8(b) shows the average time taken to mine a trajectory in the T-Drive dataset. In this experiment, we mine a trajectory only if it contains at least one anomalous edge; otherwise, based on Theorem 2 it is guaranteed to be non-anomalous. MANTRA is up to three orders of magnitude faster. Even for trajectories of size 1000 and beyond, MANTRA consumes less than 25 ms. When the anomalous edges are spread out across the trajectory, Sliding Window processes almost the same number of sub-trajectories as Naïve. Hence, Sliding Window does not show much speed-up over Naïve. Another indicator of the efficiency of MANTRA is the number of edges processed. Given a trajectory, every time an edge in the trajectory is accessed by MANTRA or Sliding Window, it is considered "processed". Fig. 8(c) shows the results. As expected, the trend is similar to that of the running time.

Since the real driver of the running time is the number of anomalous edges in a trajectory instead of its size, we next investigate the growth rate of running time against this factor. Fig. 8(d) shows the results. As against trajectory size, MANTRA is more than three orders of magnitude faster than Sliding Window and Naïve. However, the growth rates of running times for both MANTRA and Sliding Window are different than the trend observed against trajectory size. In Sliding Window, the running time decreases when the number of anomalous edges in a trajectory increases. Looking deeper into these trajectories reveals that an increase in the number of anomalous edges does not correlate with increase in trajectory size. These anomalous edges are contiguous, which often corresponds to high traffic jams. In this scenario, when the number of anomalous edges goes up, the resulting MASs cover a significant portion of the overall trajectory. As analyzed in Sec. 4.3.1, this is the exact scenario where Sliding Window starts becoming effective. Hence, we see a fall in its running time with an increase in the number of anomalous edges. For MANTRA, there is a hump in its growth rate. When the number of anomalous edges is small, the seed set converges quickly since there is less scope of interaction between impact regions. As the number of anomalous edges grows, the number of seeds goes up and consequently, the number of interacting impact regions increases resulting in higher number of iterations for seed set convergence. However, when the number of anomalous edges in a trajectory exceeds even further, the number of seeds again starts to fall since many of the anomalous edges become contiguous and group together to form a single seed.

Fig. 8(e) further substantiates efficiency of MANTRA by evaluating its performance on number of edges processed. Trends are similar to Fig. 8(d). When number of anomalous edges is beyond 400, Sliding Window processes less number of edges than MANTRA. Since when the entire trajectory, or a significant portion, is anomalous, identifying islands is a redundant operation.

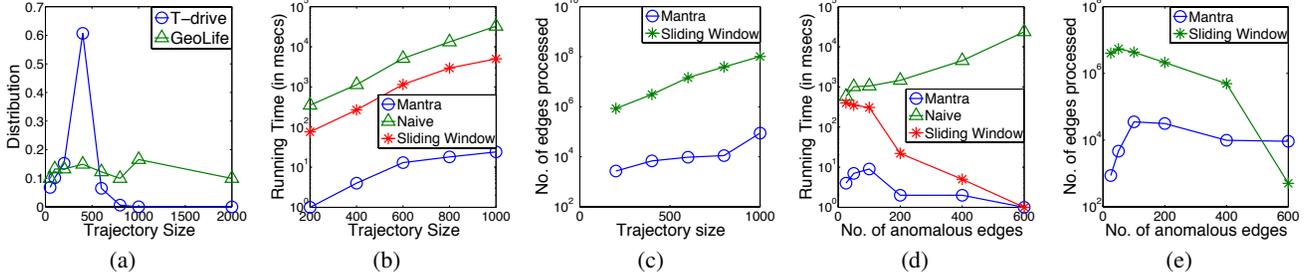Experiments against the number of anomalous edges, indirectly

Figure 8: (a) Distribution of trajectory sizes in the T-Drive and GeoLife datasets. The growth rate of (b) running time and (c) number of edges processed against the trajectory size. The growth rate of (d) running time and (e) number of edges processed against the number of anomalous edges.

show how the distribution of anomalous edges affects the running time. Consider a trajectory T, having 30% of its edges as anomalous edges. In two extreme cases, these 30% edges could either be evenly spread or grouped together as a single seed. Sliding Window's running times would drastically differ for these two cases. In next two experiments we observe the impact of this distribution directly. To have full control over the distribution of anomalous edges, only for this experiment, we synthetically alter the T-drive dataset. We randomly pick 10000 trajectories from T-Drive and alter their speeds to make them either anomalous or non-anomalous. The trajectory generation procedure uses two parameters:

• **Anomaly Percentage(AP):** AP regulates what percentage of a trajectory's edges is anomalous.

• **Seed Size Percentage(SS):** SS controls the size of the embedded seeds in a trajectory T. For example, in trajectory of length 100 if $AP = 40\%$, then 40 edges are to be set as anomalous. Now, setting $SS = 20\%$ means the size of each seed is $40 \times 0.2 = 8$. This also means there will be $40/8 = 5$ seeds that are evenly spaced out across the trajectory. Now, it is easy to see that keeping AP constant, if we set $SS = 40\%$, the seeds would be larger and hence, more of the anomalous edges would be grouped together rather than being distributed across the trajectories. Our goal in the following experiments is to verify how the running time varies with the distribution of the anomalous edges in terms of their locations.

We first fix $SS = 15\%$ and then vary AP from 5% to 50%. Results are shown in Fig. 9(a). For Sliding Window, the number of edges processed falls with AP. Even though an increase in AP translates to increase in anomalous edges, it also increases the size of the resultant MASs. As already explained, when the MAS sizes are large, Sliding Window is more effective and that behavior translates to the shown plot. MANTRA also improves with increase in AP, which shows that mining problem is more difficult when anomalous edges are sparse and evenly distributed. For MANTRA, the job is harder at lower AP, since there is more scope of interactions between impact regions of seeds and consequently, the seed set consumes more iterations to converge. This also explains why we saw a similar dip in Fig 8(d) and Fig. 8(e) against the number anomalous edges. We further study the impact of seed distribution by fixing $AP = 20\%$ and varying SS from 5% to 100%. Fig. 9(b) shows the results. As expected, when $SS = 100\%$ both algorithms process least edges. With increase in $SS\%$, the seeds are likely to be grouped together leading to quick convergence of seed sets.

Finally, we evaluate the growth rate of the running time against the anomaly threshold θ. The running time of Naïve decreases with increase in the anomaly threshold since the number of anomalous sub-trajectories decreases and consequently, the check for maximal anomalousness is faster. For Sliding Window, the running time increases with anomaly threshold. As already discussed, Sliding Window performs better when the majority of the edges in a trajectory is anomalous. This happens at lower thresholds. For MANTRA, a hump is visible and this is due to the reasons explained earlier on how it performs best at the two extreme scenarios: extremely few anomalous edges, or high portion of anomalous edges.

5.2 Analysis of the anomaly model

In Mantra, we estimate the travel time distribution in a sub-trajectory by aggregating the distributions in its constituent edges. In this section, we evaluate the accuracy of these estimations. For this purpose, we identify the 500 most frequent sub-trajectories of size X, with values of X ranging from 3 edges to 15 edges. We could not go beyond 15 edges since the number of trajectories covering a specific sequence of 16 edges or more is rare. In fact, this issue of data sparsity is the reason behind the need to approximate travel-time distribution from the constituent edges.

To set up our experiment, for each of the frequent sub-trajectories, we compute the mean and the standard deviation of the travel times. Let us denote them as μ^* and σ^*. Next, we compute the approximated mean and standard deviation, μ and σ respectively, from the edges within the sub-trajectory using the formula given in section 4.1. The accuracy of the approximation is computed as follows.

$$Accuracy(\mu) = 1 - \frac{|\mu^* - \mu|}{\mu^*} \tag{13}$$

Accuracy for σ is defined analogously. Higher the accuracy, better is the performance, with an accuracy of 1 signifying no error in approximation. Figs. 9(d) and 9(e) show the results in both T-drive and the three different classes of vehicles in the GeoLife dataset. Accuracy is above 0.9 for μ across all sub-trajectory sizes. The result is similar for σ as well. As expected, the accuracies are slightly better for sub-trajectory sizes below 5 since we assume covariance to be 0 for edges more than 5 hops apart. More importantly, there is almost no drop in accuracy beyond the size of 7.

5.3 Efficacy and applications

Beyond mathematical modeling, in this section, we evaluate the accuracy of MANTRA from the human view point; *are we able to identify sub-trajectories that would be considered anomalous by humans?* To answer this question, we perform *trajectory classification*. We use GeoLife dataset for our experiments here. GeoLife contains labeled trajectories belonging to classes Car, Bus, and Walk. We hypothesize that if a model is learned on Walk, then trajectories of other classes should be considered anomalous.
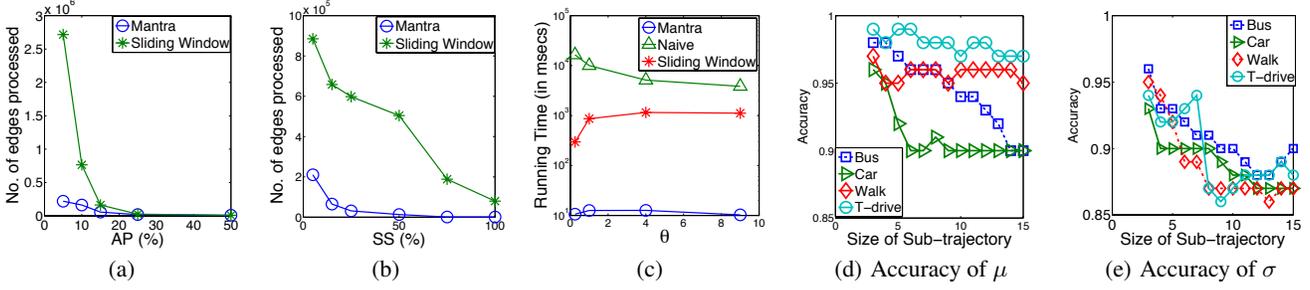
Figure 9: (a-b) Impact of the distribution of seeds on the performance of MANTRA and Sliding Window. (c) The growth rate of running time against the anomaly threshold. (d-e) The accuracy of mean μ and standard deviation σ in our approximations.

Class label	Walk	Bus
Car	0.85	0.74
Walk	-	0.75

Table 2: F-scores in two-class classification.

Class combination	car	walk	bus
Car-Walk-Bus	0.62	0.69	0.47

Table 3: F-scores in 3-class classification.

First, we verify this intuition through trajectory classification. Towards that, we perform 5-fold cross validation on GeoLife. Given a training dataset containing trajectories from k classes, we first learn the background model $\mathcal{B}_i$ for each class. Now, in the test set, for each trajectory we mine the set of MASs $\mathbb{M}_i$ corresponding to the background model of class i. Based on the mined patterns, the trajectory is mapped to a k-dimensional space where the score corresponding to dimension (or class) i is the following.

$$f_i(T, \mathcal{B}_i) = |\{e \in T \mid \exists S \in \mathbb{M}_i, e \in S\}| \qquad (14)$$

f_i essentially computes the number of edges in T that are part of some MAS in T when mined against the background model of class i. The class label $\mathcal{L}(T)$ is therefore

$$\mathcal{L}(T) = \arg \min_{1 \leq i \leq k} f_i(T, \mathcal{B}_i) \qquad (15)$$

To evaluate the performance of the classification algorithm, we perform 5-fold cross validation with balanced class labels. The accuracy of the classification is quantified using F-score [22] and shown in Table 2. As it can be seen, the F-scores are above 0.74 for all class pairs, and thus showcasing that MANTRA conforms to human intuition.

To investigate the separation between class labels in greater detail, Figs. 10(a) and 10(b) plot the anomaly scores of each trajectory in the Car-Walk and Car-Bus classification problems respectively. In these plots, each axis shows the anomaly score (Eq. 14) with respect to the denoted class. Higher the score, the more anomalous a trajectory is with that class. Majority of the walk trajectories have a high anomaly score with Car and are therefore on the right side of the plot. In contrast, although the Bus trajectories are separable from Car, their anomaly scores in comparison to Car are not as drastic as Walk. Thus, the middle zone has a significant number of Bus trajectories. This result is not surprising since discriminating between Car and Bus trajectories is difficult even for humans.

We also evaluate the performance of MANTRA in 3-class classification. In Table 3, each column shows the F-score when the corresponding class is considered as the positive class; trajectories from the remaining classes constitute the negative class. In three-class classification, a random classifier would achieve an F-score of 0.33. Against this backdrop, the performance of MANTRA is significantly higher across all combinations. This result shows that MANTRA is robust even under multiple classes.

Using the same hypothesis of anomaly, we further evaluate MANTRA and highlight its application in *trajectory segmentation*. Given a trajectory that is a mixture of multiple movement activities, temporally anomalous sub-trajectories can be mined to automatically annotate them. Consider a person who walks 1 KM to the bus stop, and then catches a bus to reach the university. By learning appropriate background models in a supervised setting, we can mine anomalous sub-trajectories and tag the various portions of the journey with the correct travel mode. Such annotated trajectory data is useful in infrastructure management and urban planning.

First, we construct trajectories as a mixture of two modes of travel. Specifically, we find trajectories of two different modes, such as car and bus, that both contain at least one common edge. We arbitrarily select one such edge. Let us denote the first trajectory as $T = [e_1, \cdots, e_i, x, \cdots, e_m]$ and the second trajectory as $T' = [e'_1, \cdots, x, e'_j, \cdots, e'_n]$, where x is the selected common edge. From these two, we construct a merged trajectory $T_m = [e_1, \cdots, e_i, x, e'_j, \cdots, e'_n]$. Our goal is now to annotate the sub-trajectory $S = [e_1, \cdots, e_i]$ with the label of T and $S' = [x, e'_j, \cdots, e'_n]$ with the label of T'.

The trajectory segmentation pipeline is setup in a manner similar to classification. We assume a training dataset that contains trajectories corresponding to each possible class. We learn a background model $\mathcal{B}_i$ for each class i in the training set. Only difference in trajectory segmentation from classification is that the test set contains merged trajectories over a mixture of two classes. Thus, instead of classifying entire trajectory, we assign a label to each edge.

Ideally, if a sub-trajectory belongs to class i, then it should be anomalous to all other classes in the training set. However, that does not occur always in real-life. A stationary car stuck in an unexpected congestion would be anomalous even in the background model of a car. Such cases need to be taken care of. First, given a trajectory T from the test set, we mine the MASs with respect to each possible class. Any sub-trajectory S that is anomalous to all classes but one, is labeled with the non-anomalous class. For the ambiguous sub-trajectories, which is either non-anomalous with multiple classes, or anomalous with all classes, we extract the set of possible class labels. $possibleClass(S)$ contains classes with which S is non-anomalous, or if S is anomalous with all classes, then $possibleClass(S)$ contains all classes. Now, label for S is

$$\mathcal{L}(S) = \arg \min_{i \in possibleClass(S)} \{\|dist_i(S)\|\} \qquad (16)$$
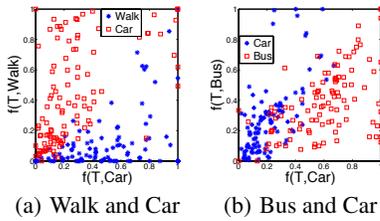
(a) Walk and Car   (b) Bus and Car

Figure 10: Anomaly scores distribution in two-class classification.

Class label	Walk		Bus	
	MANTRA	CA	MANTRA	CA
Car	0.80	0.66	0.65	0.35
Walk	-	-	0.76	0.62

Table 4: F-scores in trajectory segmentation by MANTRA and Contiguous Anomaly (CA) classification based on edge speed.

$dist_i(S)$ (Eq. 6) is deviation from background model of class i.

An obvious, alternative approach to segmentation is to classify each edge in a trajectory based on how close the speed is to the average speed of Car, Bus and Walk trajectories and report contiguous stretches of anomalous edges. We compare the performance MANTRA with this Contiguous Anomaly (CA) based segmentation approach. Table 4 shows the results. The F-score is computed in terms of how many edge labels are predicted correctly. MANTRA performs significantly better since it analyses at a sub-trajectory level. On the other hand, in CA, each edge is classified independently. Thus, if a car slows down at a traffic signal, CA classifies it as Walk. Overall, these results show that our anomaly model indeed follows the human intuition, is robust, and can be applied for effective trajectory classification and segmentation.

6. RELATED WORK

Anomaly detection on trajectory datasets can broadly be grouped into two categories: spatial anomalies [2,11,20] and temporal anomalies [12]. In [11] authors proposed a partition-and-detect framework to identify spatially outlying sub-trajectories based on density and distance. [7] proposed a spatial anomaly detection framework for evolving trajectories, using direction and density of the trajectories. To detect fraudulent taxi drivers, authors in [20] presented an anomaly detection framework for fixed source-destination pairs. In [2], a window based approach is proposed to mine anomalies in trajectory streams where a sub-trajectory inside a window is anomalous if its distance is significantly higher than its left and right neighbors. Anomaly detection framework presented in [3], detects taxi routes that deviate significantly from history. Several distance functions exist [4,5,16,17] to match trajectories based on their spatial similarity and could be used for spatial anomalies.

In temporal anomaly mining, existing studies look into global network-level anomalies, such as unexpected congestion in a road network, instead of trajectory-level anomalous sub-trajectories that we have studied in our work. [12] use historical similarity trends to identify temporal outliers, using a temporal neighborhood vector. However, none of these works addresses the particular aspect of temporal anomaly (over-speed or under-speed) on network constrained trajectories. Time series clustering has been used to detect temporal anomalies on different fields such as e-commerce [1] and speech recognition [6] and DNA optimization [18]. Clustering on time-series streams has been studied by in [15].

7. CONCLUSION

In this paper, we studied the problem of mining temporal maximal anomalous sub-trajectories from a given trajectory. We showed that complexity of the problem is quadratic with respect to trajectory size and therefore cannot be operationalized for real-time results. To overcome this bottleneck, we developed *MANTRA*. MANTRA derives its power by iteratively refining the search space into a set of "islands". These islands guarantee that all maximal anomalous sub-trajectories reside only inside them and thus the remaining part of the trajectory can be discarded. The expensive sub-trajectory level analysis is performed only within these islands to identify the answer set. Extensive experiments on real trajectories showed MANTRA as more than 3 orders of magnitude faster than baseline techniques. In addition, MANTRA achieves excellent accuracy in identifying anomalies and unleashes the ability to classify and segment trajectories based on their vehicle types.

8. REFERENCES

[1] S. Aghabozorgi and Y. W. Teh. Stock market co-movement assessment using a three-phase clustering method. *Expert Systems with Applications*, 41(4):1301–1314, 2014.

[2] Y. Bu, L. Chen, A. W.-C. Fu, and D. Liu. Efficient anomaly monitoring over moving object trajectory streams. In *KDD*, pages 159–168, 2009.

[3] C. Chen, D. Zhang, P. S. Castro, N. Li, L. Sun, and S. Li. Real-time detection of anomalous taxi trajectories from gps traces. In *Mobile and Ubiquitous Systems: Computing, Networking, and Services*. 2012.

[4] L. Chen, M. T. Özsu, and V. Oria. Robust and fast similarity search for moving object trajectories. In *SIGMOD*, pages 491–502, 2005.

[5] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh. Querying and mining of time series data: Experimental comparison of representations and distance measures. *Proc. VLDB Endow.*, 1(2):1542–1552, 2008.

[6] S. Fong. Using hierarchical time series clustering algorithm and wavelet classifier for biometric voice classification. *BioMed Research International*, 2012.

[7] Y. Ge, H. Xiong, Z.-h. Zhou, H. Ozdemir, J. Yu, and K. C. Lee. Top-eye: Top-k evolving trajectory outlier detection. In *CIKM*, 2010.

[8] C. M. Grinstead and J. L. Snell. *Introduction to Probability*. July 1997.

[9] M. R. Hustim and M. Isran. The vehicle speed distribution on heterogeneous traffic: Space mean speed analysis of light vehicles and motorcycles in makassar-indonesia. The Eastern Asia Society for Transportation Studies, 2013.

[10] V. Kolar, S. Ranu, A. P. Subramainan, Y. Shrinivasan, A. Telang, R. Kokku, and S. Raghavan. People in motion: Spatio-temporal analytics on call detail records. In *COMSNETS*, pages 1–4, 2014.

[11] J.-G. Lee, J. Han, and X. Li. Trajectory outlier detection: A partition-and-detect framework. In *ICDE*, pages 140–149, 2008.

[12] X. Li, Z. Li, J. Han, and J.-G. Lee. Temporal outlier detection in vehicle traffic data. In *ICDE*, pages 1319–1322, 2009.

[13] S. Mitra, S. Ranu, V. Kolar, A. Telang, A. Bhattacharya, R. Kokku, and S. Raghavan. Trajectory aware macro-cell planning for mobile users. In *INFOCOM*, 2015.

[14] L. Moreira-Matias, C. Ferreira, J. Gama, J. Mendes-Moreira, and J. de Sousa. Bus bunching detection by mining sequences of headway deviations. In *Advances in Data Mining. Applications and Theoretical Aspects*, volume 7377 of *Lecture Notes in Computer Science*, pages 77–91. 2012.

[15] T. Rakthanmanon, E. J. Keogh, S. Lonardi, and S. Evans. Time series epenthesis: clustering time series streams requires ignoring some data. In *ICDM*, pages 547–556, 2011.

[16] S. Ranu, P. Deepak, A. D. Telang, P. Deshpande, and S. Raghavan. Indexing and matching trajectories under inconsistent sampling rates. In *ICDE*, 2015.

[17] M. Vlachos, D. Gunopoulos, and G. Kollios. Discovering similar multidimensional trajectories. In *ICDE*, 2002.

[18] X.-H. Yang and Y.-Q. Li. Dna optimization threshold autoregressive prediction model and its application in ice condition time series. *Mathematical Problems in Engineering*, 2012, 2011.

[19] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang. T-drive: driving directions based on taxi trajectories. In *SIGSPATIAL GIS*, 2010.

[20] D. Zhang, N. Li, Z.-H. Zhou, C. Chen, L. Sun, and S. Li. ibat: Detecting anomalous taxi trajectories from gps traces. In *UbiComp*, 2011.

[21] Y. Zheng, X. Xie, and W.-Y. Ma. Geolife: A collaborative social networking service among user, location and trajectory. *IEEE Data Eng. Bull.*, 33(2):32–39, 2010.

[22] S. Zhou, S. Zhang, and G. Karypis. *Advanced Data Mining and Applications: 8th International Conference, ADMA 2012, Nanjing, China, December 15-18, 2012, Proceedings*. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012.