Real-time Detection of Anomalous Taxi Trajectories from GPS Traces

Chao Chen[1], Daqing Zhang[1], Pablo Samuel Castro[1], Nan Li[23], Lin Sun[1], and Shijian Li[4]

[1] Institut TELECOM/TELECOM SudParis, CNRS SAMOVAR, France
{chao.chen,daqing.zhang,pablo.castro,lin.sun}@it-sudparis.eu
[2] National Key Laboratory for Novel Software Technology, Nanjing University, China
lin@lamda.nju.edu.cn
[3] School of Mathematical Sciences, Soochow University, China
[4] Department of Computer Science, Zhejiang University, China
shijianli@zju.edu.cn

Abstract. Trajectories obtained from GPS-enabled taxis grant us an opportunity to not only extract meaningful statistics, dynamics and behaviors about certain urban road users, but also to monitor adverse and/or malicious events. In this paper we focus on the problem of detecting anomalous routes by comparing against historically "normal" routes. We propose a real-time method, *iBOAT*, that is able to detect anomalous trajectories "on-the-fly", as well as identify which parts of the trajectory are responsible for its anomalousness. We evaluate our method on a large dataset of taxi GPS logs and verify that it has excellent accuracy (AUC ≥ 0.99) and overcomes many of the shortcomings of other state-of-the-art methods.

1 Introduction

With the increasing pervasiveness of GPS devices, there is an enormous amount of information available to researchers [24]. The traces left behind by GPS-enabled vehicles provide us with an unprecedented window into the dynamics of a city's road network. This information has been analyzed to uncover traffic patterns [17], city dynamics [27], driving directions [23], and a city's "hot-spots" [5, 28]. Much of this work has made use of the data from GPS-equipped taxis, often using it to provide useful information for the taxi drivers themselves [5, 19, 20].

Mining large GPS traces has been investigated for a number of different problems, and one interesting amongst these is using GPS traces to develop new ways to detect taxi drivers' anomalous activities. One kind of anomaly could be caused by greedy taxi drivers, who aim to overcharge passengers by deliberately taking unnecessary detours. It would be useful if we could detect the anomalous behavior while it is occurring as well as which parts of the trajectory (sub-trajectories) are abnormal, thereby reducing the number of passengers that fall prey to taxi fraud. Another anomalous situation could occur when there are abnormal traffic conditions such as traffic accidents, resulting in certain road segments being blocked, forcing taxi drivers to find alternate routes. Real-time traffic monitoring of blocked road segments can be achieved through the real-time detection of anomalous sub-trajectories.

The aforementioned problem is the focus of this paper, where we aim to detect driving routes that are considered *anomalous* in the sense that they differ significantly from the norm. Given that one of the main motivations for this work is the detection of fraudulent taxi drivers, the anomaly detection will be with respect to fixed source and destination *areas.* Accurate detection of these anomalous driving patterns can be useful for detecting adverse traffic events, road network changes and taxi fraud, amongst others. We believe successful methods should be able to identify which parts of a trajectory are anomalous, as well as assign an *ongoing anomaly score* which can be used to rank the different trajectories. Our main contributions are a real-time method which accurately identifies anomalous sub-trajectories with very little processing overhead, as well as computes an evolving anomaly score which can indicate how severely the anomalous route deviates from normal routes. We begin by reviewing related work in Section 2. The algorithm is defined in Section 3, and an empirical evaluation along with an analysis of its differences with a closely related method are presented in Section 4. Finally, we present concluding remarks and point to future research directions in Section 5.

2 Related work

There have been many recent works on mining large GPS traces. Liao, et al. [15] devise methods to predict a user's mode of transportation and daily routine to provide reminders when needed, while [19, 12] uncover taxi drivers' operating patterns. Other works show how to predict the route and destination based on historical GPS traces [10, 6, 29], in addition to providing driving directions by exploiting taxi drivers' knowledge [23]. GPS traces have also been used for uncovering interesting "hot-spots" for tourists [28, 26], for passengers searching for vacant taxis [20], or for classifying the social functions of different regions in a city [21].

Anomaly detection has its roots in the more general problem of outlier detection. In most cases the data is static, rather than evolving over time. There are a number of different methods available for outlier detection, including supervised approaches [1], distance-based [2, 9], density-based [3], model-based [8] and isolation-based methods [18].

Recent work on detecting anomalous moving vehicles include the following. In [11], a trajectory is split into various partitions (at equal intervals) and a hybrid of distance and density based approaches is used to classify each partition as anomalous or not. In [7], the authors compute a score based on the evolving moving direction and density of trajectories, and make use of a decay function to include previous scores. In [4], Bu et al. present a method for monitoring anomalies over continuous trajectory streams by using the local continuity characteristics of trajectories to cluster trajectories on a local level; anomalies are then identified by a pruning mechanism. Somewhat related, but addressing a different problem, Li et al. [14] identify outlier road segments by detecting drastic changes between current data and historical trends. Finally, some recent work has used learning methods to identify anomalous trajectories [13, 16, 22]. However, these last methods require training data which is expensive to label.

Most of these methods identify anomalous trajectories based on their physical distance to "normal" clusters or their orientations. Based on the idea of isolating anomalies [18], Zhang et al. [25] devise a method which identifies trajectories as anomalous when they follow paths that are rare with respect to historical trajectories. Our paper adopts this characterization of anomalous trajectories but goes a step further: in addition to identifying anomalous trajectories online, our method is able to specify which *parts* of the trajectory are anomalous. In comparison with some of the more sophisticated methods mentioned above, whose running time may disqualify them from real-time situations, our method is fast and can be used in a real-time manner.

3 iBOAT: Isolation Based Online Anomaly Trajectory Detection

Definition 1. *A* **trajectory** *t consists of a sequence of points $\langle p_1, p_2 \ldots, p_n\rangle$, where $p_i \in \mathbb{R}^2$ is the physical location (*i.e. *latitude/longitude). We will use t_i to reference position i in t, and for any $1 \leq i < j \leq n$, $t_{i\to j}$ denotes the* **sub-trajectory** *$\langle p_i, \ldots, p_j\rangle$.*

The points p_i occur in a continuous domain, so dealing with them directly is difficult. In order to mitigate this problem, we assume we have access to a finite decomposition of the area of interest. Specifically, let G be a finite set of elements, and $\rho : \mathbb{R}^2 \to G$ a function that maps locations to elements in G.

Definition 2. *A* **mapped trajectory** *$\bar{t}$, obtained from a trajectory t, consists of a sequence of elements $\langle g_1, g_2 \ldots, g_n\rangle$, where for all $1 \leq i \leq n$, $g_i \in G$ and $\bar{t}_i = \rho(t_i)$. We will write $g \in \bar{t}$ when $\bar{t}_i = g$ for some $1 \leq i \leq n$.*

Henceforth we will only deal with mapped trajectories, so we will drop the *mapped* qualifier. Let $\mathbb{T}$ denote the set of all mapped trajectories. Define the function $pos : \mathbb{T} \times G \to \mathbb{N}^+$, where given a trajectory t and element g returns the first index in t that is equal to g (or ∞ if $g \notin t$).

$$pos(t, g) = \begin{cases} \arg\min_{i\in\mathbb{N}^+}\{t_i = g\} & \text{if } g \in t \\ \infty & \text{otherwise} \end{cases}$$

For example, if $t = \langle g_3, g_4, g_5, g_4, g_3, g_6\rangle$, then $pos(t, g_4) = 2$, and $pos(t, g_7) = \infty$.

Given a fixed source-destination pair with a set of trajectories T and a sub-trajectory $t = \langle g_1, g_2, \ldots, g_n\rangle$, we would like to verify whether t is **anomalous** with respect to T. We say a sub-trajectory t is anomalous with respect to T (and

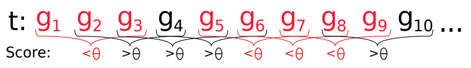

Fig. 1. Example of fixed-window approach with $k = 3$. There are two anomalous sub-trajectories detected: $\{g_1, g_2, g_3\}$ and $\{g_5, g_6, g_7, g_8, g_9\}$.

the fixed source-destination pair) if the path it follows rarely occurs in T. We define a function $hasPath : \mathcal{P}(\mathbb{T}) \times \mathbb{T} \rightarrow \mathcal{P}(\mathbb{T})$ that returns the set of trajectories from T that contain all of the points in t in the correct order.[1] Note, however, that the points need not be *sequential*, it suffices that they appear in the same order.

$$hasPath(T, t) = \left\{ t' \in T \,\middle|\, \begin{array}{l} \text{(i) } \forall 1 \leq i \leq n.\ g_i \in t' \\ \text{(ii) } \forall 1 \leq i < j \leq n.\ pos(t', g_i) < pos(t', g_j) \end{array} \right\} \quad (1)$$

For instance, if $T = \{t1, t2, t3\}$, where $t1 = \langle g_1, g_2, g_3, g_4, g_5, g_8, g_9, g_{10} \rangle$, $t2 = \langle g_1, g_2, g_4, g_5, g_6, g_8, g_{10} \rangle$, and $t3 = \langle g_1, g_3, g_4, g_3, g_6, g_8, g_{10} \rangle$, and an ongoing trajectory $t = \langle g_1, g_2, g_5, g_8 \rangle$, then $hasPath(T, t) = \{t1, t2\}$.

We now formalize what it means for a sub-trajectory t to be anomalous with respect to a set T by observing the proportion of trajectories agreeing with t.

Definition 3. *Given a threshold* $0 \leq \theta \leq 1$*, a sub-trajectory* t *is* **θ-anomalous** *with respect to a set of trajectories* T *if*

$$support(T, t) = \frac{|hasPath(T, t)|}{|T|} < \theta \quad (2)$$

We will use this definition of θ-anomalousness to describe two variants of our proposed algorithm.

Fixed-window: We fix a window size k, indicating the number of grid cells in the trajectory to check for anomalousness. Given a set of trajectories T and an ongoing trajectory $t = \langle g_1, g_2, \ldots \rangle$, we verify whether the last k-sized sub-trajectory from t occur with enough frequency in T to determine if it is anomalous. An example of this approach is illustrated in Figure 1, where $k = 3$. Note that when $k = 1$, we have the density method used for comparison in [25].

Adaptive: In this approach we maintain a *working set* of trajectories (initially equal to T). After i entries received, our partial trajectory t consists of $\langle g_1, g_2, \ldots, g_i \rangle$ and we have a working set T_i. Upon arrival of entry g_{i+1}, we compute $support(T_i, t)$. If its value is less than θ, then point g_{i+1} is anomalous so it is added to the set of anomalous points, and we set $T_{i+1} = T$; otherwise, we set $T_{i+1} = hasPath(T_i, t)$. This procedure is repeated as long as new entries are arriving. Note that $T_0 = T$, and that every time an anomalous point is encountered, the working set is reset to the original trajectory set T. See Figure 2 for an illustration of this process. This resetting is what enables our adaptive algorithm to accurately detect anomalous sub-trajectories in real-time with a finer granularity than the fixed-window approach (with $k > 1$). Additionally, by reducing the working set with each incoming point, the adaptive approach has a computational advantage over the fixed-window approach.

[1] $\mathcal{P}(X)$ is the power set of X.

t:	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8	g_9	g_{10} …
Working set:	T_0	T_0	T_0	T_1	T_2	T_3	T_0	T_0	T_1'	T_2'
Working set size:	$>\theta$	$<\theta$	$<\theta$	$>\theta$	$>\theta$	$>\theta$	$<\theta$	$<\theta$	$>\theta$	$>\theta$

Fig. 2. Example of adaptive window approach. Two anomalous sub-trajectories are detected: $\{g_2, g_3\}$ and $\{g_7, g_8\}$. Note that $T_1 = hasPath(T_0, \langle g_3, g_4 \rangle)$ and $T_1' = hasPath(T_0, \langle g_8, g_9 \rangle)$.

Anomaly score: We assign an ongoing score based on the length of the anomalous sections so far. Let $dist : G \times G \to \mathbb{R}$ be a distance function on the elements of G (this will usually be the standard Euclidean distance). Given a trajectory t along with a set of anomalous points, we define the *score* as the sum of the distance between all anomalous points g_i and the previous point g_{i-1} from t. The detected anomalous points in Figure 2 are $\{g_2, g_3, g_7, g_8\}$, then the score will be the sum of $dist(g_1, g_2)$, $dist(g_2, g_3)$, $dist(g_6, g_7)$ and $dist(g_7, g_8)$.

4 Empirical evaluation

For our experiments, we make use of a large database of GPS logs from over 7600 taxis in Hangzhou, China. Each log contains the latitude, longitude and taxi status (free/occupied), amongst other things. The logs were collected over a period of twelve months at a sample rate of around one entry per minute. In this work we will only make use of trajectories where the taxi is occupied, as one of the applications of this method is in aiding passengers to avoid fraud. We restrict our attention to the Hangzhou metropolitan area, with longitude and latitude ranges of $[120.0°E, 120.5°E]$ and $[30.15°N, 30.40°N]$, respectively. We decompose the area just mentioned into a matrix of 100×200 grid cells (*i.e.* $|G| = 20000$), where the area of each grid cell is roughly $250m^2$. Thus, the function ρ simply maps a latitude/longitude pair into the grid cell g enclosing it.

Because of the rate at which GPS entries are received and the small size of our grid cells, there may be gaps between consecutive mapped points (black squares in Figure 3). We augment all the trajectories to ensure that there are no gaps in the trajectories by (roughly) following the line segment (green line in left panel) between the two cells in question (gray cells). When we are testing whether a trajectory t is anomalous, even though it may be following the same

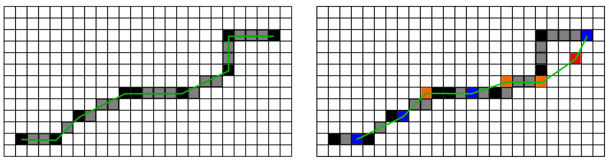

Fig. 3. Left: An example of a trajectory with augmented cells. Right: Comparing existing trajectory with a new trajectory.

path as most of the trajectories in T, the GPS points may fall in different cells. In the right panel of Figure 3 we display the augmented trajectory from the left panel, along with a new testing trajectory (colored squares and green line). Some of the grid cells fall on the augmented path (blue squares), while others fall in "empty" grid cells (orange and red cells). There is the possibility that the augmented path was not completely accurate, so we must account for this type of error when testing with a new trajectory: If a grid cell of the new trajectory is adjacent to one of the augmented cells, we consider it as if it were along the same path (orange cells), while if it is not adjacent to any augmented cells, we consider it as following a different path (red cell).

We picked nine source-destination pairs (T-1 through T-9) that had sufficient trajectories between them (at least 450, but on average over 1000), and asked three volunteers to manually label whether trajectories are anomalous or not (averaged at around 5.1% over the nine datasets). We then only labeled trajectories as anomalous if it received a majority of "votes" from the volunteers.

4.1 Results

To test *iBOAT*, we selected a trajectory t as an ongoing trajectory from a dataset T and used our two approaches with $\theta = 0.05$. This was done for all trajectories and all datasets. In the left panel of Figure 4 we display the output of our method for two test trajectories from T-1, where we plot the normal trajectories in light blue; for the test trajectories, the anomalous points are drawn in red and the rest (normal points) in dark blue. As can be seen, our method can accurately detect which parts of a trajectory are anomalous and which are normal. In the right panels of Figure 4 we plot $support(T - \{t\}, t)$ (see equation (2)) for the ongoing trajectory t. We can see that the value of *support* is a clear indication of when trajectories become anomalous, and that there is little difference between the different variants of *iBOAT*. Note, however, that there is a trailing *lag* for the fixed-window approach, equal to k. This is because the last anomalous point in an anomalous sub-trajectory will be included in the following k sub-trajectories. Although setting $k = 1$ will solve the lag problem, this minimal window size contains no contextual information of the trajectory, and will therefore have poor prediction quality. This was observed in [25] (therein referred to as the density method), and will be evident in the figures below. A classified trajectory will fall into one of four scenarios: True Positive (TP), when an anomalous trajectory is

	T-1	*T-2*	*T-3*	*T-4*	*T-5*	*T-6*	*T-7*	*T-8*	*T-9*
iBAT	0.9868	0.9970	0.9970	0.9909	0.9944	0.9997	0.9983	0.9972	0.9998
$k = 1$	0.9629	0.9364	0.8023	0.8518	0.9108	0.9227	0.8806	0.9380	0.9788
$k = 2$	0.9904	0.9900	0.9735	0.9582	0.9887	0.9914	0.9846	0.9735	0.9989
$k = 3$	0.9805	0.9890	0.9386	0.9571	0.9879	0.9899	0.9841	0.9728	0.9986
Adaptive	0.9982	0.9952	0.9962	0.9890	0.9967	0.9953	0.9935	0.9936	0.9995

Table 1. AUC values of the different algorithms.

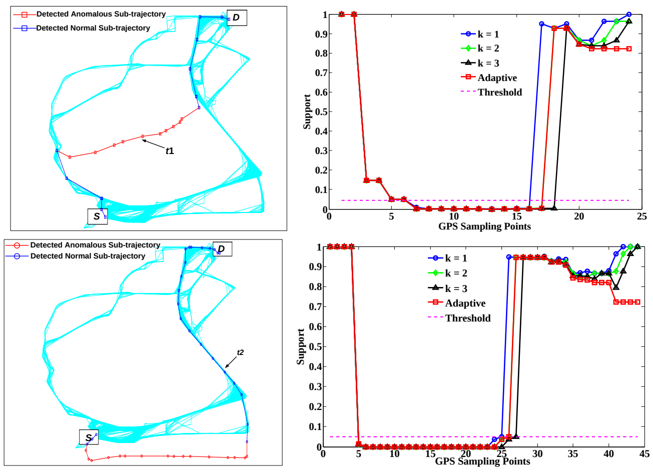

Fig. 4. Left: Detected anomalous sub-trajectories from T-1 using *iBOAT*. Right: Plot of ongoing *support*.

correctly classified as anomalous; False Positive (FP), when a normal trajectory is incorrectly classified as anomalous; False Negative (FN), when an anomalous trajectory is incorrectly classified as normal; True Negative (TN), when a normal trajectory is correctly classified as normal. The True Positive Rate (TPR), defined as $TPR = \frac{TP}{TP+FN}$, measures the proportion of correctly labeled anomalous trajectories; the False Positive Rate (FPR), defined as $FPR = \frac{FP}{FP+TN}$, measures the proportion of *false alarms* (*i.e.* normal trajectories that are labeled as anomalous). A perfect classifier will have $TPR = 1$ and $FPR = 0$. In a ROC curve, we plot FPR on the x-axis and TPR on the y-axis, which indicates the tradeoff between false alarms and accurate classifications. By measuring the Area Under Curve (AUC), we can quantify the tradeoff between correct positive classification and false alarms. In Figure 5 we plot the ROC-curve for T-1 and T-8, and in Table 1 we display the AUC values for all datasets and the different algorithms. To generate this plot we ranked all the instances according to the scores from each algorithm and used the *perfcurve* function in MATLAB, which generates the ROC curve. Using *iBAT*, $k = 2$ and the adaptive variants of *iBOAT* have the best overall performance with little significant difference between them. In Section 4.3 we will discuss some important differences between adaptive *iBOAT* and the fixed-window and *iBAT* approaches.

4.2 *iBOAT* versus *iBAT*

iBAT is a recent anomaly detection method introduced in [25] that is similar to our approach. In order to determine whether a trajectory is anomalous, *iBAT* picks cells from the testing trajectory at random to split the collection of trajectories into those that contain the cell and those that do not. This process is repeated until the trajectory is isolated, or until there are no more cells in the

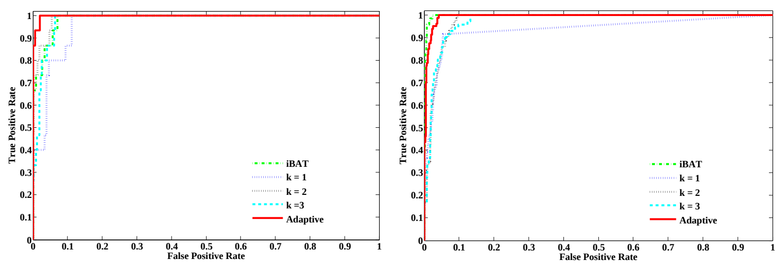

Fig. 5. The ROC curves for T-1 (left) and T-8 (right).

trajectory. Usually the number of cells required to isolate anomalous trajectories will be much less than the number of cells in the trajectory. This isolation procedure is repeated a number of times and $\mathbb{E}(n(t))$, the average number of cells required to isolate a trajectory, is used to compute the score, which is proportional to $2^{-\mathbb{E}(n(t))}$.

Our proposed method is a clear improvement over *iBAT* on two levels. First of all, we are able to determine which *parts* of a trajectory are anomalous, in contrast to *iBAT* which only classifies *full* trajectories as anomalous. Second of all, our method works in *real-time*: we can detect anomalous sections immediately, and do not require a full trajectory as an input.

In Figure 6 we show an example where a road block has forced a taxi to retrace its path and search for another route to its destination. We focus on the first part of the trajectory where the taxi retraces its steps. In the right panel of Figure 6 we can see the *support* is accurately identifying the anomalous section of the trajectory. We determined what anomalous ranking (based on the scores) both methods assign this partial trajectory in comparison with all other trajectories[2]. Out of 1418 trajectories, *iBOAT* ranked this trajectory in 48th place, while *iBAT* ranked it in 831th place. Furthermore, *iBAT* assigned this trajectory a score of 0.4342, which is below the usual 0.5 threshold. Thus, while *iBAT* is unable to detect that this trajectory is anomalous, *iBOAT* has ranked it amongst the top 3% of anomalous trajectories, as well as identifying which part is anomalous. The reason *iBAT* fails in this example is that their method does not take the *order* the points appear in into consideration; despite the fact that the taxi is retracing its steps and actually going *away* from the destination, it is only visiting "normal" grid cells.

Now consider the hypothetical example in Figure 7. In this simple situation, the value $\mathbb{E}(n(t))$ for *iBAT* is just the expected number of times their algorithm must pick cells before an anomalous cell (in red) is picked. This is essentially a Bernoulli trial with "success" probability p equal to the proportion of anomalous cells to total number of cells in the trajectory. It is well known that the expected number of trials before reaching success in a Bernoulli trial is given by $1/p$. Let n be the number of cells in the straight line between S and D, then trajectories of the form on the left will have $2n - 2$ anomalous cells and $5n - 4$ total cells,

[2] A higher ranking means higher degree of anomalousness.

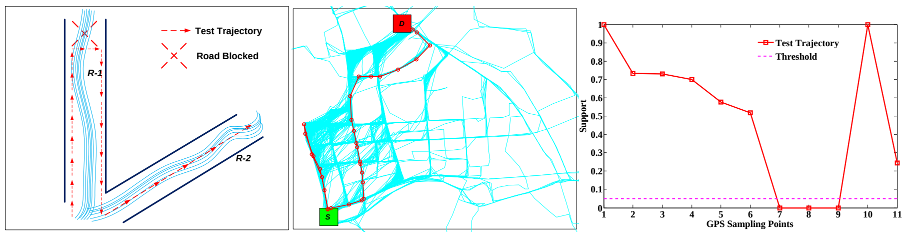

Fig. 6. A trajectory where the taxi had to retrace its path due to a blocked route. Left: illustration of situation; Middle: Real trajectory; Right: Ongoing *support* from *iBOAT*.

while trajectories of the form on the right will have $2n - 2$ anomalous cells and $2n+2$ total cells. It follows that for trajectories of the form on the left $\mathbb{E}(n(t)) = \frac{5n-4}{2n-2} \rightarrow \frac{5}{2} \Rightarrow score \approx 0.1768$; for trajectories of the form on the right $\mathbb{E}(n(t)) = \frac{2n+2}{2n-2} \rightarrow 1 \Rightarrow score = 0.5$. Thus, *iBAT* will qualify trajectories of the form on the right as more anomalous than those on the left. This runs contrary to intuition, which would perceive trajectories like the one on the left at least as anomalous as the one on the right, given that the path taken is much longer and they are clearly taking longer routes than necessary. Our scoring method, which uses the distance of the anomalous sub-trajectories, would assign the left trajectory an anomalous score around 33% higher than the one on the right. Finally, we compared the running time of both algorithms on all the datasets, and we display the results in Figure 8. We computed the running time for checking each trajectory in each dataset, and averaged over the size of the dataset. Although *iBAT* will usually check fewer grid cells than *iBOAT* (since one anomalous cell is enough to classify the trajectory as anomalous), *iBAT* is based on random cell selections, so they must average over m runs; as in [25], we set $m = 50$. We can see that *iBOAT* is consistently faster than *iBAT* on all datasets.

4.3 Adaptive versus fixed-window approach

As was evident in the previous figures, the performance of the fixed-window approach with $k = 2$ and the adaptive approach are nearly identical. The advantage of the fixed-window approach is that it requires a very small amount of

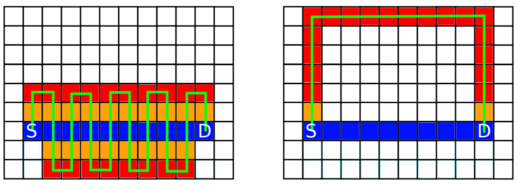

Fig. 7. Two anomalous trajectories of different types. The normal trajectory between S and D is in blue, cells adjacent to normal cells are in orange, and anomalous cells in red.

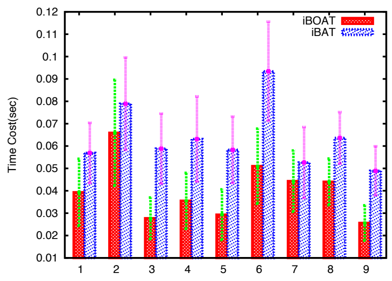

Fig. 8. Running times of *iBOAT* and *iBAT* on all the datasets.

memory for real-time anomalous detection, while the adaptive method requires memory proportional to the size of the longest "normal" sub-trajectory. In practice, this difference is negligible. On the other hand, we will demonstrate that the adaptive approach has an advantage over the fixed-window approach due to its use of longer historical "contexts". In Figure 9 we display an anomalous trajectory that "switches" from one normal route to another. The fixed-window method with $k = 2$ will not detect this anomalous switch. The transition from point 19 to point 20 will seem normal since this sequence occurs in route A, and the transition from point 20 to point 21 will also seem normal since it occurs in route B. On the other hand, *iBOAT* would maintain the entire route up to the point when the driver switches routes and would immediately detect it as an anomalous point. Although this example is specific to window sizes equal to 2, similar situations (with longer overlaps between routes) will produce a similar effect for larger window sizes.

5 Discussion and Conclusion

In this paper we have proposed a new algorithm for fast *real-time* detection of anomalous trajectories obtained from GPS devices that can use fixed and variable window sizes. In addition to classifying full trajectories as anomalous,

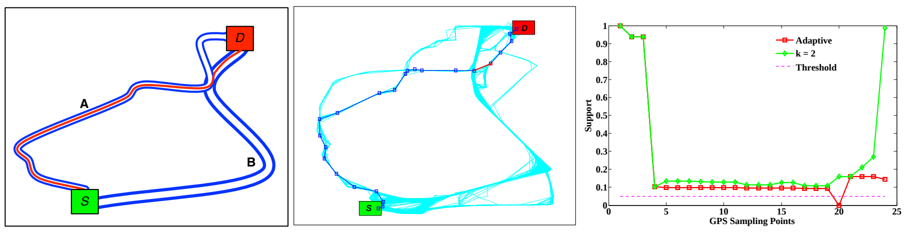

Fig. 9. A situation the fixed-window method ($k = 2$) fails to classify as anomalous: two normal routes (route A and B) are in dark blue; an anomalous trajectory (in red) switches from route A to route B at their intersection. Left: illustration of situation; Middle: Real trajectories; Right: ongoing *support* from *iBOAT*.

iBOAT can work with ongoing trajectories and can determine which parts of a trajectory are responsible for its anomalousness.

We validated *iBOAT* on a large dataset of taxi GPS trajectories recorded over a month and found our method achieved excellent performance (AUC$\geq$ 0.99 for all datasets) which is comparable to *iBAT*'s performance; however, we demonstrated a number of examples that highlight *iBOAT*'s advantage over *iBAT* and the sliding window method.

Given that one of the main applications of this work is taxi fraud detection, information such as speed can be crucial for more accurate detection. We plan on extending our work to use additional attributes such as speed, distance, orientation, taxi fare, etc., as this information can help distinguish fraudulent detours from other types of anomalous trajectories (such as road closures). The work presented in this paper was not meant to distinguish the different types of anomalous trajectories, but is an important first step in this direction. We have recently constructed a digital map of Hangzhou along with a mechanism for mapping GPS points onto the digital map; we are currently investigating *iBOAT*'s performance on this map and preliminary results are promising. This would give us access to detailed information about possible accidents, road closures, etc., which would be of great benefit to drivers, passengers, and traffic monitoring organizations.

Acknowledgements

The authors would like to thank the three anonymous reviewers for their valuable comments as well as Prof. Zhi-hua Zhou for the precious inputs. Chao Chen would like to thank the Chinese Government for his PhD funding. This research was supported by the Institut TELECOM "Futur et ruptures" Program, NSF of China (No.60975043, No.60803109), and the High-Tech Program of China (863) (No.2011AA010104).

References

1. N. Abe, B. Zadrozny, and J. Langford. Outlier detection by active learning. In *Proc. KDD*, 2006.
2. F. Angiulli and F. Fassetti. Dolphin: An efficient algorithm for mining distance-based outliers in very large datasets. *ACM-TKDD*, 3(1), 2009.
3. M.M. Breunig, H.P. Kriegel, R.T. Ng, and J. Sander. LOF: Identifying density-based local outliers. In *Proc. SIGMOD*, 2000.
4. Y. Bu, L. Chen, A.W.C. Fu, and D. Liu. Efficient anomaly monitoring over moving object trajectory streams. In *Proc. KDD*, 2009.
5. H. Chang, Y. Tai, H. Chen, and J.Y. Hsu. iTaxi: Context-aware taxi demand hotspots prediction using ontology and data mining approaches. In *Proc. of TAAI*, 2008.
6. J. Froehlich and J. Krumm. Route prediction from trip observations. In *Proc. SAE*, 2008.

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 *Proc. CIKM*, 2010.
8. Z. He, X. Xu, and S. Deng. Discovering cluster-based local outliers. *Pattern Recognition Letters*, 24(9–10):1641–1650, 2003.
9. E.M. Knorr, R.T. Ng, and V. Tucakov. Distance-based outliers: Algorithms and applications. *VLDB Journal*, 8(3-4):237–253, 2000.
10. J. Krumm and E. Horvitz. Predestination: Inferring destinations from partial trajectories. In *Proc. UBICOMP*, 2006.
11. J. Lee, J. Han, and X. Li. Trajectory Outlier Detection: A Partition-and-Detect Framework. In *Proc. ICDE*, 2008.
12. B. Li, D. Zhang, L. Sun, C. Chen, S. Li, G. Qi, and Q. Yang. Hunting or waiting? Discovering passenger-finding strategies from a large-scale real-world taxi dataset. In *PerCom Workshops*, 2011.
13. X. Li, J. Han, S. Kim, and H. Gonzalez. ROAM: Rule- and motif-based anomaly detection in massive moving object data sets. In *Proc. SDM*, 2007.
14. X. Li, Z. Li, J. Han, and J.G. Lee. Temporal outlier detection in vehicle traffic data. In *Proc. ICDE*, 2009.
15. L. Liao, D.J. Patterson, D. Fox, and H. Kautz. Learning and inferring transportation routines. *Artificial Intelligence*, 171(5–6):311–331, 2007.
16. Z. Liao, Y. Yu, and B. Chen. Anomaly detection in GPS data based on visual analytics. In *Proc. VAST*, 2010.
17. M. Lippi, M. Bertini, and P. Frasconi. Collective traffic forecasting. In *Proc. of ECML-PKDD: Part II*, 2010.
18. F.T. Liu, K.M. Ting, and Z. Zhou. Isolation Forest. In *Proc. ICDM*, 2008.
19. L. Liu, C. Andris, and C. Ratti. Uncovering cabdrivers' behavior patterns from their digital traces. *Computers, Environment and Urban Systems*, 34:541–548, 2010.
20. S. Phithakkitnukoon, M. Veloso, A. Biderman, C. Bento, and C. Ratti. Taxi-Aware Map: Identifying and predicting vacant taxis in the city. In *Proc. AmI*, 2010.
21. G. Qi, X. Li, S. Li, G. Pan, and Z. Wang. Measuring Social Functions of City Regions from Large-scale Taxi Behaviors. In *PerCom Workshop*, 2011.
22. R. Sillito and R.B. Fisher. Semi-supervised learning for anomalous trajectory detection. In *Proc. BMVC*, 2008.
23. J. Yuan and Y. Zheng. T-Drive: Driving Directions Based on Taxi Trajectories. In *ACM SIGSPATIAL GIS*, 2010.
24. D. Zhang, B. Guo, and Z. Yu. The Emergence of Social and Community Intelligence. *IEEE Computer*, 44(7):21–28, 2011.
25. D. Zhang, N. Li, Z. Zhou, C. Chen, L. Sun, and S. Li. iBAT: Detecting Anomalous Taxi Trajectories from GPS Traces. In *Proc. of UbiComp*, 2011.
26. V.W. Zheng, B. Cao, Y. Zheng, X. Xie, and Q. Yang. Collaborative filtering meets mobile recommendation: A user-centered approach. In *Proc. AAAI*, 2010.
27. Y. Zheng, Y. Liu, J. Yuan, and X. Xie. Urban Computing with Taxicabs. In *Proc. of Ubicomp*, 2011.
28. Y. Zheng, L. Zhang, X. Xie, and W. Ma. Mining interesting locations and travel sequences from gps trajectories. In *Proc. WWW*, 2009.
29. B.D. Ziebart, A.L. Maas, A.K. Dey, and J.A. Bagnell. Navigate Like a Cabbie: Probabilistic Reasonoing from Observed Context-Aware Behavior. In *Proc. of Ubicomp*, 2008.