Avoidance Region Discovery: A Summary of Results

Emre Eftelioglu* Xun Tang* Shashi Shekhar*

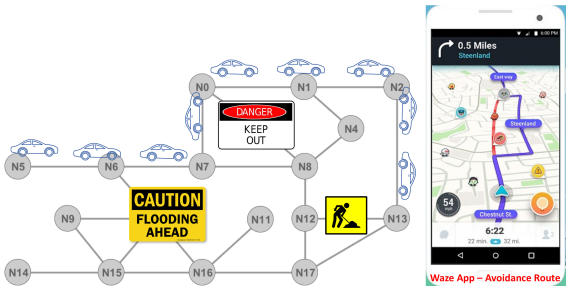
Abstract

Given a set of GPS trajectories, avoidance region discovery (ARD) finds regions that are avoided by drivers. ARD is important for applications such as sociology, city/transportation planning and crime mitigation, where it can help domain users understand the driver behavior under different concerns (e.g. rush hour, congestion, dangerous neighborhood, etc.). ARD is challenging because of the large number of trajectories with thousands of GPS points, large number of candidate avoidance regions, and the cost of evaluating those. Related work is focused on finding evasive trajectories for a given set of avoidance regions. Distinct from the related work, we propose an Avoidance Region Miner (ARM) approach that can detect both the avoidance regions and evasive trajectories just by using the trajectories in hand without the need of an additional input. A case study on real trajectory data confirms that ARM discovers such regions for further investigation by domain users. Experiments show that ARM yields substantial computational savings compared to a baseline approach.

1 Introduction

Informally, given a collection of GPS trajectories and a road network, avoidance region discovery (ARD) finds the regions drivers tend to avoid. ARD has important applications in transportation/city planning, sociology and criminology. For example, city planners may focus on detecting avoidance regions to identify the underlying reasons causing drivers bypass them. Figure 1(a) illustrates such problems (e.g. construction, flood, danger) that may cause avoidance instead of using a shorter or faster path. Figure 1(b) shows a mobile application, i.e. Waze [4], which uses crowd-sourcing to provide paths that do not intersect dangerous neighborhoods. However, Waze relies on user-contributed information whereas our work automates the discovery of avoidance regions, based strictly on available trajectory data. Thus, ARD has fewer data dependencies than Waze. Also, ARD will discover regions that are avoided by the general population and criminals, while Waze is not designed to discover any such regions.

1.1 Application Domain: In sociology, domain scientists work to identify neighborhoods with specific



(a) (b)

Figure 1: An example of possible conditions causing drivers' avoidance behavior (left). Waze application allows users to prevent driving in high crime rate regions (right).

demographic structures. Identifying regions that are avoided by drivers may help investigate the underlying reasons. In fact, most cities have some distressed neighborhoods that are known to be riskier than other neighborhoods [4]. For example, in the Chinese city of Kunming, taxi drivers try to avoid getting customers from the regions where marginalized people are thought to be living [19]. Finding those regions may help city planners have new investments as well as policy changes to improve the public opinion. Note that some of such distressed regions may already be known by the officials. However, our approach aims to find both the ones already known and emerging before they are noticed by the majority. For instance, trajectories from taxis over a year may help detect the emergence of such distressed regions before they are well known by all drivers.

Using ARD to investigate traffic patterns, transportation planners can identify reasons for congestion during rush hours [5] or road segments victimized by potholes/cracks that endanger the safety of motorists.

In criminology, to preserve anonymity, criminals tend to avoid security checkpoints and security cameras [15]. The trajectories of this behavior may generate avoidance regions. Finding these regions, which also have security checkpoints/cameras may lead to the detection of such individuals. Similarly, some criminals may do a surveillance of a target site (e.g. banks) before they start their criminal action (e.g. robbery). These surveillance visits may lead to circling/triangulating the region by driving around the block causing an avoidance behavior. Identifying those locations may help public security officials prevent such crimes before happening.

*Department of Computer Science, University of Minnesota.

1.2 Challenges: ARD is challenging due to large number of candidate avoidance regions (CAR), spatial networks (e.g. 10^6 edges in a road network) and trajectories (TR) (e.g. 10^5 trajectories in PlanetGPX [2]).

In ARD, CAR are the interior faces of the road network because these correspond to the city block polygons where buildings and neighborhoods (e.g. sets of city blocks) are located. Assuming the road network is planar, Euler's theorem [21] states that the total number of faces is $|faces|+|N|-|E|=2$. Since $|faces|$ includes an exterior-infinite face, $|CAR| = |faces|-1 = |E| - |N| + 1$, where $|E|$ is the number of edges (e.g. street segments) and $|N|$ is the number of nodes (e.g. road intersections). Computing those faces has a time complexity of $O(|E|log|E|)$.

For each $car \in CAR$, we need to evaluate at what extent they are avoided. This is a challenging task because we need a comparison with trajectories to understand the avoidance behavior. Since we lack the information about the driver preferences for different routes (due to many factors affecting those choices), we used shortest paths for such comparison. Thus, shortest paths for each trajectory are generated increasing the cost by $O(|TR| \times (|N| + |E|) \times log|N|)$.

Next, the avoidance polygons (AP), i.e. the regions between the trajectory and shortest path pairs, are generated. This requires a comparison between each edge of every shortest path and trajectory pair. Since, there are thousands of trajectories with tens of edges, this phase is also challenging.

Finally, each $ap \in AP$ is needed to be compared with each $car \in CAR$ as well as each shortest path to determine the avoidance and non-avoidance behaviors causing an exorbitant computational cost.

1.3 Related Work: Patterns of evasion in trajectory data is an important task in trajectory mining. Recent work in this area includes finding anomalous behavior of taxi trajectories [27, 6] by discretizing the study area into grids and representing each trajectory using grid cells and comparing them with a normally-behaving trajectory grid representation; finding top-k trajectory outliers [10] using their grid representations; clustering trajectories to find similar movement behaviors [30]; inferring transportation modes (e.g. walking, driving, etc.) as a classification problem [31, 29, 22, 28]; finding "more preferred" routes using drivers' behavior [23, 32, 24]; and investigating human mobility patterns [7, 11, 26, 20, 25, 9]. These works are interesting for identifying interesting movement patterns and provide insights for future trips. They are not concerned, however, with the identification of regions that exhibit particular movement patterns, i.e. avoidance.

Figure 2 classifies avoidance region discovery in two

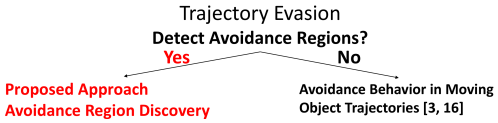


Figure 2: Related Work.

groups namely candidate avoidance regions are given and candidate avoidance regions are inferred from trajectories themselves. In the first category, the avoidance regions are given by the users [3, 16] and thus known beforehand. Our work studies avoidance regions that may be unknown. In fact, it can be hard to predetermine such regions due to different reasons causing avoidance behavior (concerns about safety, reluctance to go through security checkpoints, discomfort in distressed neighborhoods, etc.). In addition, the related work [3] consider trajectories as segments consisting of GPS point pairs, which may lead to biased results when GPS points are collected from a noisy environment. Finally, it does not account for underlying road structure.

In contrast, the proposed approach aims to identify regions of avoidance by comparing trajectories with their shortest paths rather than with user input (Figure 2 left branch). Moreover, trajectories are considered as a whole starting from the first GPS point to the last. Finally, the underlying road network is considered, which provides better approximation of real world phenomena. To the best of our knowledge, the proposed approach is the first of its kind for identifying avoidance regions from GPS trajectories.

1.4 Contributions: (i) We define avoidance region discovery (ARD) problem and (ii) propose a baseline algorithm to solve the ARD. In order to improve the scalability of the baseline algorithm, (iii) we propose an Avoidance Region Miner (ARM) algorithm that leverage the algorithmic structure of the baseline algorithm. (iv) Results are qualitatively (i.e. via case study) and quantitatively (i.e. via experiments) evaluated to show the superiority of ARM over the baseline approach.

1.5 Scope and Outline of the Paper: This paper focuses on finding evasive patterns modeled as avoidance regions where each face in a road network graph is considered as a candidate. Trajectories are map-matched to provide their edge representation on a spatial network and then compared with their corresponding shortest paths (i.e. same source and destination). (i) When there are multiple shortest paths with the same cost, the shortest path with the highest number of edges in common with the trajectory (i.e. shares the most edges with a trajectory) is used. We assume that such a shortest path is unique (see the discussion in Section 6). (ii) Proposed approach aims to infer avoidance regions. Since a fair comparison cannot be done

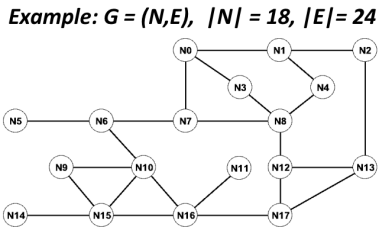


Figure 3: Spatial network with 18 nodes and 24 edges.

with approaches (e.g.[3]) where users input avoidance regions, case studies and experiments were conducted only with the algorithms proposed in this paper. (iii) Parameter selection is out of the scope and this paper does not provide guidance for parameters. However, one may set the input threshold (i.e. λ) to its lowest value (i.e. $\lambda = 0$) and investigate all output patterns.

The paper is organized as follows: Section 2 provides the key definitions and the problem statement. The proposed approach is covered in Section 3. Section 4 presents a case study showing the output of ARD with a real world trajectory dataset. The experimental evaluation is covered in Section 5. Discussions about ARD are in Section 6. Section 7 concludes the paper and gives an overview of future work.

2 Definitions and Problem statement

2.1 Definitions:

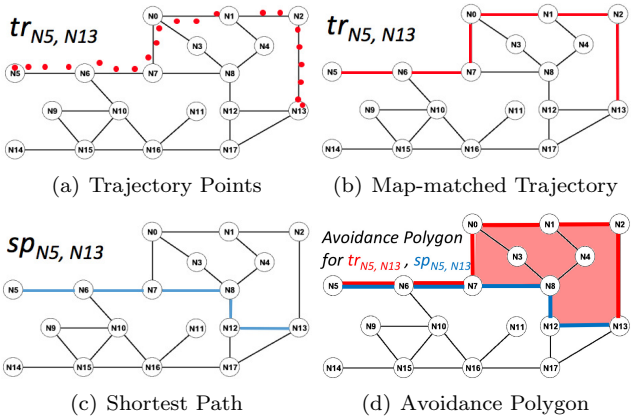
DEFINITION 1. *A **spatial network (e.g. road network)** $G = (N, E)$ is a set of nodes (N) and edges (E) where each node $n_v \in N$ is associated with coordinates (x, y) in an Euclidean space. E is a subset of the cross product $N \times N$ and an edge $e_i \in E$, which joins nodes n_u and n_v, is associated with a length and speed limit representing its travel time $w_{u,v} \geq 0$.*

For example, Figure 3 shows a spatial network with $|N| = 18$ and $|E| = 24$.

DEFINITION 2. *A **trajectory** is a set of chronologically ordered GPS points ($p_{1...m}$), where each point consists of a spatial coordinate set and a time stamp represented as $p_j = (x, y, t)$. Trajectories are map-matched [18, 14] to provide their edge representation in the spatial network. Thus, $tr_i = e_s \rightarrow e_2 \rightarrow ... \rightarrow e_d$*

For example, the trajectory in Figure 4(a) consists of 21 points. The map-matched trajectory in Figure 4(b) illustrates its edge representation in G.

DEFINITION 3. *A **shortest path** $sp_{u,v}$ is a sequence of nodes $[n_1, n_2, ..., n_i] \in N$ such that $[e_1, e_2, ..., e_i] \in E$ and $n_i \in N$ are distinct and the sum of the travel time w is minimized. Thus, the cost of a shortest path is $W_{sp} = \sum w_{e \in sp}$. For multiple "sp"s with the same cost, the sp assumed to be the one with the highest number of common edges with the trajectory (see Section 6).*



(a) Trajectory Points (b) Map-matched Trajectory

(c) Shortest Path (d) Avoidance Polygon

Figure 4: An example trajectory (4(a)), its map-matched edge representation(4(b)), corresponding shortest path(4(c)) and this pair's avoidance polygon(4(d)).

In Figure 4(c), the shortest path is shown in blue.

DEFINITION 4. ***An Avoidance Polygon*** *($ap \in AP$) is a polygon bounded by the nodes in the map-matched trajectory (tr_i) and its shortest path (sp_i). The ap is a bounded region, and reflects the additional cost of travel to avoid the part of the sp not common to the tr. By providing a measure of the frequency with which an avoidance polygon's constituent "car"s are being avoided, we will focus on regions where drivers are willing to travel with higher costs to avoid the shortest path and the corresponding ap. Note that for a single tr and sp pair, there may be multiple ap.*

In Figure 4(d), the *ap* (red) is created by the differing edges of the *tr* (Figure 4(b)) and *sp* (Figure 4(c)).

DEFINITION 5. ***Candidate Avoidance Regions*** *($car \in CAR$) are the polygons that are interior faces of the spatial network. Using Euler's theorem for planar graphs [21], the number of candidate avoidance regions ($|CAR|$) in a spatial network is $|CAR| = |E| - |N| + 1$.*

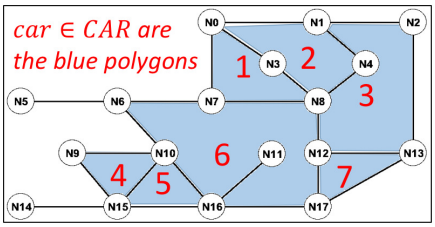


Figure 5: Example candidate avoidance regions.

For example, in Figure 5 the total number of candidate avoidance regions ($|CAR|$) is 7 since $|E| = 24$ and $|N| = 18$. Thus, $|CAR| = 24 - 18 + 1 = 7$.

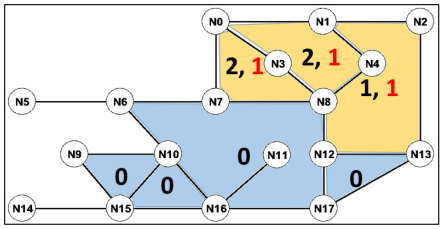


Figure 6: Avoidance (black) and Non-avoidance (red) counts of "*car*"s. Yellow polygons have $c > 0$.

DEFINITION 6. ***Avoidance Count*** *(c) of a* $car_i \in CAR$ *is the number of times the* car_i *is covered by (9I model in topology [8]) an* $ap_j \in AP$. *Suppose* T *denotes the subset of avoidance polygons that cover a car, thus* $T = \{ap_j | ap_j \in AP \text{ and } car_i \subseteq ap_j\}$ *and* $c_{car_i} = |T|$.

DEFINITION 7. ***Non-Avoidance Count*** *(nc) of a* car_i *aims to determine how many trajectories (except the one that created it) are* **expected** *to intersect an* ap_j. *Thus, it is the count of* $sp_k \in SP$ *that intersect the* $ap_j \in T$ *(T denotes the "ap"s that cover the* car_i *as defined above) which cover the* car_i. *Let* $S = \{sp_k | sp_k \in SP \text{ and } \exists\ ap_j \in T \text{ such that } ap_j \cap sp_k \neq \emptyset\}$. *Now,* $nc_{car_i} = |S|$.

Given the 3 trajectories in Figure 7(b), the avoidance (black) and non-avoidance counts (red) of each candidate avoidance region are shown in Figure 6. The yellow polygons have $c > 0$.

DEFINITION 8. ***Interestingness Ratio (I):*** *Interestingness ratio is the expression of the drivers' willingness to avoid a region compared to others who do not avoid. Consider each car* $\in CAR$ *is assigned with the number of being avoided (c) and not avoided (nc). Using c and nc, their interestingness ratio is* $I = \left(\frac{c}{c+nc}\right) \times c$.

For the trajectories in Figure 7(b), the I for the red polygons in Figure 7(e) can be computed as follows: $c = 2$, $c + nc = 2 + 1$. Thus, $I = \left(\frac{2}{3}\right) \times 2 = 4/3$. Similarly, the yellow region has $I = \left(\frac{1}{2}\right) \times 1 = 1/2$.

DEFINITION 9. ***Interesting Avoidance Region*** *(iar) is a car with* $I \geq \lambda$, *where* λ *is a user defined threshold.*

2.2 Problem Statement: The avoidance region discovery (ARD) problem is formulated as follows:
Given: **(1)** a spatial network $G = (N, E)$, **(2)** a set of GPS trajectories (TR), and **(3)** an Interestingness Ratio Threshold (λ);
Find: Interesting avoidance regions with $I \geq \lambda$.
Objective: Scalability, correctness, completeness.
Constraint: When there are multiple shortest paths, trajectories are compared with the shortest paths that have the highest number of common edges and such shortest path is assumed to be unique.

For example, given the spatial network in Figure 7(a), the trajectories in Figure 7(b), and $\lambda = 1/2$, the output is depicted in Figure 7(e).

Algorithm 1 Baseline algorithm for ARD

```
Input:
    1) A spatial network with G = (N, E),
    2) A set of trajectories tr ∈ TR and
    3) Interestingness Ratio threshold (λ)
Output:
    Interesting avoidance regions with I ≥ λ.
Algorithm:
    Step 1:  Candidate avoidance region enumeration
 1: for each Unvisited Edge e_i ∈ G do
 2:    Put e_i in E_car
 3:    while e_j = GetNextClockwiseGraphWalkEdge(e_i) ≠ e_i do
 4:       e_j → E_car
 5:    car = CreatePolygon(E_car) and car → CAR
 6:    Mark e_i as Visited and E_car = ∅
    Step 2:  Map-match Trajectories (using  [18, 14])
 7: for each Trajectory tr_i ∈ TR do
 8:    tr_i = MapMatch(tr_i) and tr_i → TR_mm
    Step 3:  Compute Shortest Paths
 9: for each Trajectory tr_i ∈ TR_mm do
10:    Set start and end nodes n_s, n_e
11:    Compute sp_i using A* Algorithm [13] and sp_i → SP
    Step 4:  Create Avoidance Polygons
12: for each tr_i ∈ TR_mm and sp_i ∈ SP do
13:    for each Edge e_j ∈ tr_i and Edge e_k ∈ sp_i do
14:       if e_j ∉ sp_i then  e_i → E_ap_i
15:       if e_k ∉ tr_i then  e_k → E_ap_i
16:    ap_i = CreatePolygon(E_ap_i)
17: ap_i → AP
    Step 5-6:  Compare Avoidance Polygons with CAR and SP
18: for each car_i ∈ CAR do
19:    for each ap_j ∈ AP do
20:       if ap_j ⊆ car_i then  c_i = c_i + 1
21:       for each sp_k ∈ SP do
22:          if sp_k ⊆ ap_j then nc_i = n_i + 1
    Step 7:  Compute I and Return iars
23: for each car_i ∈ CAR do  Compute I_i
24:    if I_i ≥ λ then Return car_i as iar
```

3 Proposed Approach

3.1 Baseline Algorithm: The baseline algorithm to solve the ARD problem has seven steps (Alg. 1):
Step 1: Candidate avoidance region enumeration: The road network graph is traversed and using the edges of each empty cycle (no node or edge inside), a *car* polygon is generated and added to CAR.
Step 2: Map-match Trajectories: Next, trajectories are map-matched [18, 14] to get their edge representation on the road network.
Step 3: Compute Shortest Paths: Shortest path for each trajectory is computed using A* algorithm [13].
Step 4: Create Avoidance Polygons: Comparing sp and tr pairs, avoidance polygons (AP) are created using the edges that are not common for the pair.
Step 5-6: Compare Avoidance Polygons with CAR and SP: For each $ap \in AP$, all "*car*"s are compared to determine those that are completely inside the ap and their avoidance counts are increased by 1. Similarly, all $sp \in SP$ are compared with the ap and for each sp that intersects it, the non-avoidance counts of those "*car*"s inside ap are increased.

Step 7: Compute I and Return Interesting Avoidance Regions: For the candidate avoidance regions which have $c > 0$, Interestingness Ratios are computed and the ones which exceed the λ threshold are returned as the interesting avoidance regions.

Note that due to the large number of trajectories, avoidance polygons, and candidate avoidance regions; even the baseline algorithm uses spatial indexing, e.g. R-Tree with STR [17], to speed-up the execution time.

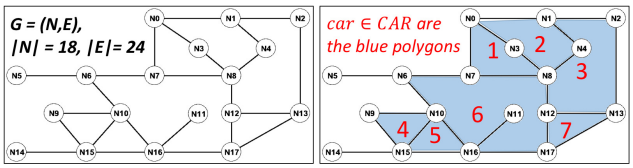
Execution Trace: Figure 7 shows an illustrative execution trace of the baseline approach. In Step 1, candidate avoidance regions are created (Figure 7(a)). Using Euler's theorem [21], $|CAR| = 24 - 18 + 1 = 7$ (shown in blue). In step 2, trajectories are map-matched to provide their edge representation (Figure 7(b)). Next, shortest paths (step 3) are computed (in blue). In step 4, avoidance polygons are generated (Figure 7(c)). In this example, out of the 3 trajectories, one is following the shortest path and does not have any avoidance polygons. However, 2 trajectories are using non-shortest paths causing them create avoidance polygons. For this step, the trajectory and the shortest path are compared node by node and if any node/edge of shortest path and the trajectory are not the same, they are kept in an array of nodes/edges. Once they re-start following the same route again, the *ap* is generated (e.g. using simple cycles from graph theory). For example, the $tr_{N5,N11}$ and $sp_{N5,N11}$ (on top of Figure 7(b)) pair differs starting from the node $N7$ and merge at node $N17$, thus the edges between $N7$ and $N17$ of both shortest path and trajectory will be used to create the *ap* on the left of Figure 7(c).

In step 5-6, the avoidance polygons (Figure 7(c)) and the candidate avoidance regions (Figure 7(a)) are compared to determine the avoidance counts (c). Yellow polygons on the left of Figure 7(d) have $c > 0$ since they are covered by the avoidance polygons of the trajectories. Similarly, the non-avoidance counts (nc) are computed using the shortest paths. Since only one trajectory is following the shortest path and intersects those avoidance polygons, the nc of yellow candidate avoidance patterns are 1. In step 7, the interestingness ratios are computed using the c and nc. For example, the red "*car*"s have $c = 2$, $c + nc = 2 + 1$. Thus, $I = \left(\frac{2}{3}\right) \times 2 = 4/3$. If λ was selected as $1/2$ the red and yellow regions would be in the output (Figure 7(e)).

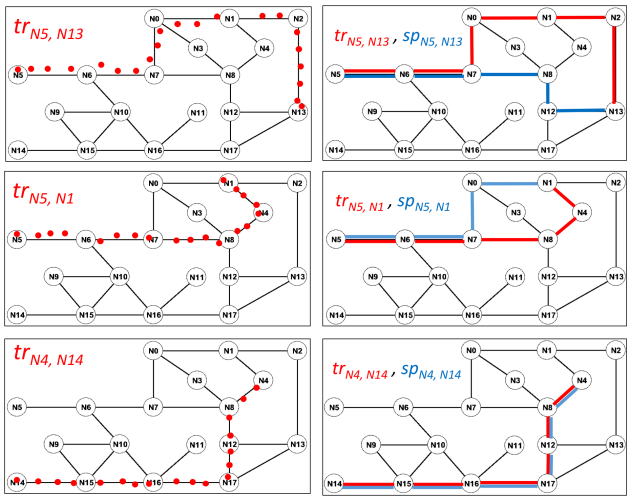
3.2 Avoidance Region Miner:

The Avoidance Region Miner (ARM), shown in Alg. 2, uses two refinements to improve computational scalability without comprising the completeness of the baseline algorithm.

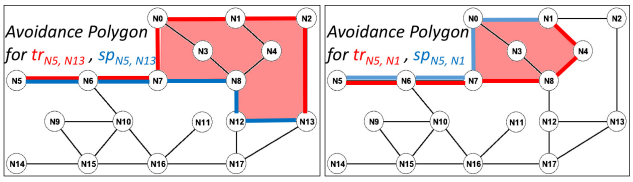
Elimination of Redundant Candidate Avoidance Region Enumeration: Our observation from the Baseline algorithm is that $car \in CAR$ are generated



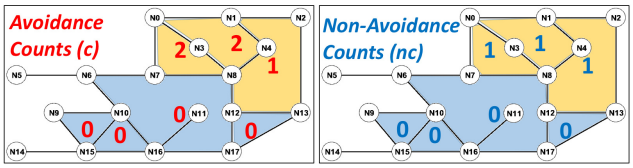
(a) Input Spatial Network and Generated "*car*"s



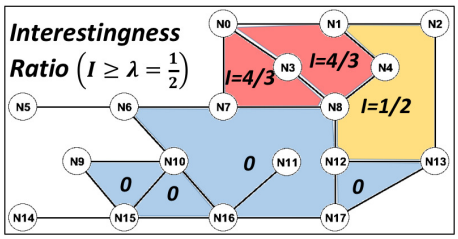
(b) Map-matched trajectories and corresponding shortest paths



(c) "*ap*"s generated for the input trajectories. $tr_{N4,N14}$ follows the shortest path and does not have an *ap*



(d) Avoidance and non-avoidance counts



(e) Output Avoidance Regions

Figure 7: Execution trace of ARD. 7(a) shows the spatial network and candidate avoidance regions (*car*). 7(b) shows input trajectories, their map-matched representations (red) and shortest paths (blue). Avoidance polygons are depicted in 7(c). 7(d) shows avoidance and non-avoidance counts for each *car*. Finally, 7(e) shows the interestingness ratio calculation and the output.

Algorithm 2 Avoidance Region Miner (ARM) algorithm

```
Input: Inputs are the same as the Baseline Algorithm
Output:
    Interesting avoidance regions with I ≥ λ.
Algorithm:
    Step 1 -3
 1: Same as the baseline algorithm
    Step 4:  Create a APS Polygon
 2: for each ap_i ∈ AP do
 3:    APS = APS ∪ ap_i
    Create sub-graph G_a
 4: for each Edge e_i ∈ G and n_i ∈ G do
 5:    if e_i ⊆ APS||n_i ⊆ APS then
 6:       Put e_i and n_i to G_a
    Step 5 - 6
 7: Same as the the baseline algorithm but with G_a
    Step 7:  Compute I and Return iars
 8: for each car_i ∈ CAR do
 9:    if c_i ≥ λ then  compute I_i
10:    if I_i ≥ λ then  return car_i
```

even if there is no avoidance polygon. For large spatial networks, where the trajectories and shortest paths are concentrated only in a smaller region in the study area, this causes a high number of candidate avoidance regions to be generated. To overcome this issue, we propose avoiding unnecessary candidate avoidance region enumeration by (1) initially creating the avoidance polygons from each trajectory and shortest path pair, (2) merging these to create a larger avoidance polygon (APS), and (3) finally getting an avoidance sub-graph (G_a) of the spatial network that is covered by (in a topological sense) APS. Thus, instead of creating "*car*"s for the whole spatial network (G), we create them for an avoidance sub-graph (G_a) of the spatial network.

DEFINITION 10. ***Avoidance Polygon Set*** *(APS), is the union of all avoidance polygons in the study area.*

$$APS = \bigcup_{i=1}^{|AP|} ap_i$$

DEFINITION 11. ***An Avoidance Sub-Graph*** *(G_a), is a sub-graph of spatial network (G), where $G_a \subset G$ and E_a and N_a are covered by APS, so $G_a \subseteq APS$.*

For example, given the trajectories and shortest paths in Figure 7(b), the APS is the union of the red polygons in Figure 7(e). The G_a nodes are $N_a = \{N0, N1, N2, N3, N4, \ N7, N8, N12, N13\}$ and edges are $E_a = \{e_{N0,N1}, e_{N1,N2}, \ ... \ e_{N1,N4}\}$. Using this subgraph, instead of creating the 7 candidate avoidance regions in Figure 7(a), we will create only the ones tagged as $1, 2,$ *and* 3 reducing the cost of enumeration.

Elimination of Unnecessary Interestingness Ratio Computation: For each *car*, the interestingness measure (I) is computed and compared against a λ threshold. The max. value of I is c when there are no trajectories not avoiding the *car*. In other words, when $nc = 0$ the $I = \left(\frac{c}{c+0}\right) \times c = c$. Thus, we eliminate computation of I for the *car* if $c < \lambda$ since these will never exceed λ threshold.

For example, given the avoidance counts in the bottom row of Figure 7, if $\lambda = 2$, we don't need to compute nc as well as I of the "*car*"s with count $c < 2$.

4 Case Study

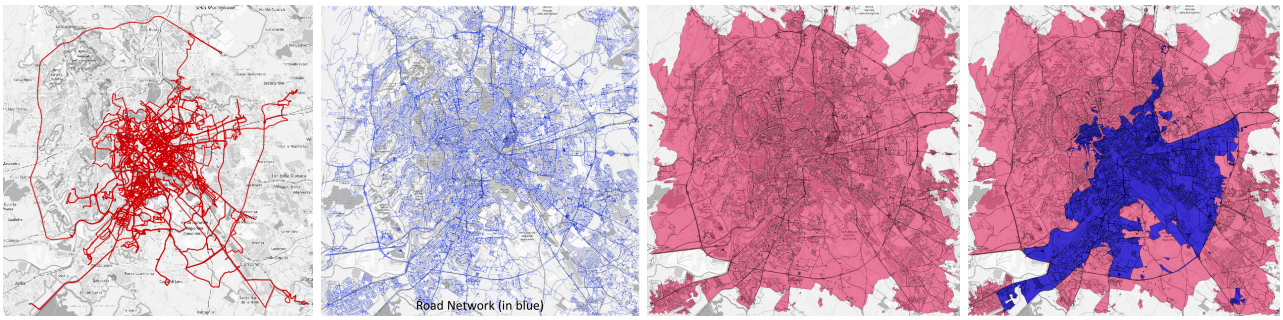
We conducted a case study on a real dataset consisting of 1312 vehicle GPS trajectories collected between February 2008 and February 2011 in Rome, Italy [2] as shown in Figure 8(a). For long trajectories spanning to many hours and days, we split them if there were stops lasting more than 5 minutes. The road network graph, extracted from Open Street Map [12], includes approximately 75000 edges and 56000 nodes as visualized in Figure 8(b). The road network graph includes 20967 candidate avoidance regions (Figure 8(c)) but with the algorithmic refinement in ARM, we created 10093 "*car*"s (Figure 8(d)) and evaluated each against $\lambda = 2$. The output interesting avoidance regions are shown in blue in Figure 8(f). Figure 8 shows that ARM identified regions that are avoided by many trajectories. In Figure 8(g), the interesting avoidance regions on the north are around the British Ambassador's residence as well as two hospitals. Drivers may be avoiding this area due to closed streets or congestion caused by increased security measures around the residence or higher than normal traffic around the hospitals. However, the interpretation of the output is left to domain users.

5 Experimental Evaluation

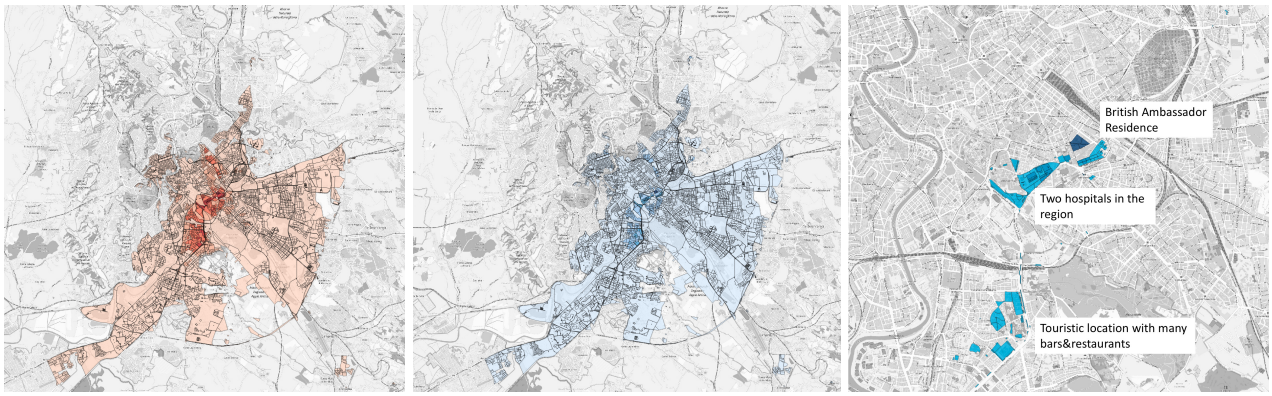
The goal of the experiments was twofold: to evaluate the performance of ARM under different parameters and to compare its performance with the baseline approach. To achieve these goals, the following questions were asked: (1) What is the effect of the number of trajectories? (2) What is the effect of the number of avoidance polygons? (3) What is the effect of the road network size? (4) What is the effect of λ threshold?

The closest work to our problem is [3], which does not detect avoidance regions but detects trajectories that avoid a given region. Thus, due to the lack of comparable work on this problem, experiments were conducted with the two proposed algorithms.

Experimental Design: Experiments were performed on the same real trajectory and road network datasets that are used for the case study. The default values for these datasets are 1000 trajectories, 56000 nodes and 75000 edges, $\lambda = 2$. For the experiments on road network size, we varied these values while preserving the connectedness of the spatial network. Thus, the number of edges and nodes were varied together. In order to observe the effect of varying number of avoidance polygons, we did not change the trajectories but randomly removed/added avoidance polygons. Both algorithms were implemented on Java platform and exe-



(a) Input Trajectory Dataset (b) Input Road Network (c) Candidate Avoidance Regions without refinement (d) Candidate Avoidance Regions with refinement (blue)



(e) Candidate Avoidance Regions colored by their avoidance counts (f) Output Interesting Avoidance Regions (g) A zoomed in view of the Interesting Avoidance Regions

Figure 8: The road network of Rome, Italy, the input trajectories, the candidate avoidance regions and the output of ARM for $\lambda = 2$. Visualizations created using QGIS Software [1] and OSM Maps [12] with 1/160000 scale.

cuted 10 times for each experiment on a MacBook Pro with an Intel Core i7 2.5 GHz CPU and 16GB memory.

5.1 Experimental Results :

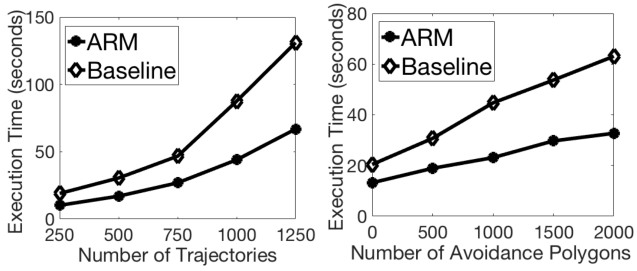
Effect of the Number of Trajectories: We varied the number of trajectories ranging from 250 to 1250. Figure 9(a) shows that the proposed ARM algorithm performs faster than the baseline approach. Although both algorithms' execution times are affected, the savings from the refinements in ARM increase when the number of trajectories increase.

Effect of the Number of Avoidance Polygons: We varied the number of avoidance polygons ranging from 0 to 5000. Figure 9(b) shows that ARM is less affected by the number of avoidance polygons than the baseline approach thanks to its refinement that eliminates unnecessary interest measure computation.

Effect of the Road Network Size: We varied the number of edges in the road network graph from 75000 to 175000 edges. To keep the graph connected, once edges are removed, non-connected nodes are also removed. To prevent incorrect map-matching of trajectories, instead of removing edges and nodes from the original road network graph, we added more nodes and edges by zooming out from the current study area to include more nodes and edges. Figure 9(c) shows that the proposed ARM algorithm performs faster than the baseline approach. The ARM algorithm is not affected by the number of edges in the spatial network graph thanks to the elimination of redundant candidate avoidance region enumeration refinement.

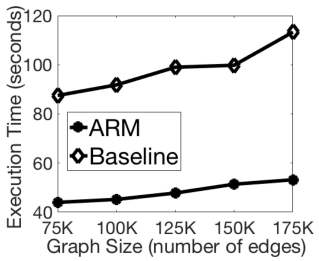
Effect of the interestingness ratio threshold: We varied the λ from 0 to 5 and observe its effect on the execution times. Figure 9(d) shows that baseline algorithm's execution time is not affected λ since it does not have an algorithmic refinement that leverage the properties of the interestingness measure. However, ARM benefits from the λ threshold and the savings increase with higher interestingness ratio thresholds.

Comparison of Algorithm Steps: To understand the bottlenecks of both algorithms, we created a bar plot with the execution times of each step. In Figure 9(e), the bottleneck of both algorithms is the last step where the counts are updated for candidate avoidance regions and the interestingness measures are computed. It can be seen that the refinements of ARM reduced the cost of this step substantially.

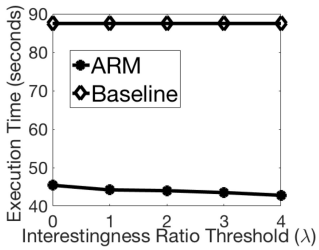


(a) Effect of $|TR|$

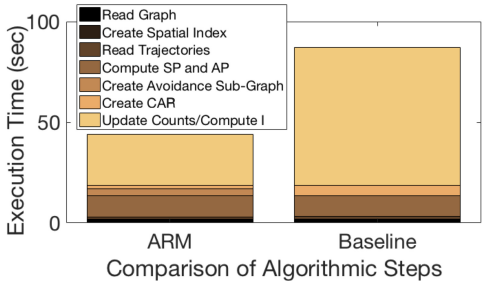
(b) Effect of $|AP|$



(c) Effect of $|E|$ and $|N|$



(d) Effect of λ



(e) Comparison of Baseline and ARM Steps

Figure 9: Experimental results for Baseline and ARM.

6 Discussion

Deviation from Shortest Path: To identify the avoidance behavior, we compared trajectories with their corresponding shortest paths. Figure 10 shows an histogram of their pairwise length differences. For example, ∼1000 out of 1312 trajectories have a length between 1 *and* 1.5× shortest path. Using this histogram, we eliminated trajectories with *major deviation* ($\geq 5\times$ longer than the shortest path) since they may not represent avoidance behavior. For example, when an individual goes to sightseeing, i.e. the trip starts and ends at the same location or very close locations, the trajectory can be hundreds times longer than the shortest path.

Minor differences between a shortest path and trajectory may also be interesting. For example, criminals may be avoiding only one city block (if there are security checkpoints), compared to drivers who may avoid a large neighborhood (with tens of city blocks). Depending on the application domain and the domain specific criteria, users should eliminate the use of trajectories that are not interesting for that specific domain. In our work, we did not eliminate such trajectories.

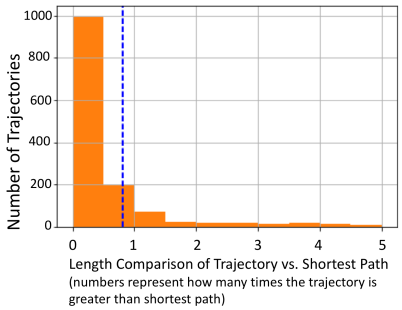


Figure 10: Histogram of the length of the trajectories compared to their shortest paths.

Shortest Path Assumption: ARM uses shortest paths for comparison with trajectories. However, people may prefer non-shortest paths due to rush hour, road congestion or routes with multiple stops (e.g. gas station, grocery shopping, etc.). However, ARM is not dependent on the shortest paths and different paths can be used to compare trajectories. In the future, behavior of drivers derived from historical trajectory data will be used to identify such avoidance behaviors.

Non-Unique Shortest Paths: For grid-like road networks (e.g. Chicago downtown with equally divided city blocks), there may be multiple shortest paths. For such conditions, we used the shortest paths with the highest number of common edges with the trajectories and assumed that such shortest paths are unique.

Global and Individual Behavior: Our datasets did not have identifiers for individuals due to privacy. For some applications (e.g. criminology), individual behaviors may be interesting (e.g. suspicious activities). With slight changes, our approach can distinguish global and individual behaviors.

Statistical Significance: We explore avoidance regions in terms of avoidance and non-avoidance counts. Yet, this may lead to the detection of chance regions. A statistical significance test along with the distribution of trajectories in the study area is required to quantify the significance of avoidance patterns. However, distribution of trajectories is hard to determine as it is affected by multiple concerns (i.e. time, trajectory length and duration). If we can overcome this challenge, we plan to incorporate a statistical significance test in the future.

7 Conclusion

We explored the avoidance region discovery (ARD) problem which has important societal applications, e.g. sociology, transportation and city planning, etc. ARD is challenging due to the large number of trajectories with thousands of GPS points, large number of candidates and the cost of evaluating those. We proposed the Avoidance Region Miner (ARM), which is the first algorithm that discovers avoidance regions and quantifies the avoidance level of each. ARM improves the scalabil-

ity of a baseline approach substantially by eliminating redundant computations. The case study provides evidence that the algorithm works, as it detected areas that drivers avoided due to closed streets or congestion caused by the security measures around the embassies or higher than normal traffic around the hospitals.

In future, we envision adding a statistical significance test to eliminate chance regions. In addition, we plan to compare the trajectories with the most used routes instead of shortest paths. Finally, we plan to use larger datasets both for experiments and case studies to identify the bottlenecks of the proposed approaches under different conditions.

8 Acknowledgments

This material is based upon work supported by the USDOD Grant No. HM1582-08-1-0017. We would like to thank Kim Koffolt and the University of Minnesota Spatial Computing Research Group for their comments.

References

[1] *Qgis software*, www. qgis. org/en/site, (2015).
[2] *Planetgpx*, 2017, `wiki.openstreetmap.org/wiki/GPX`.
[3] L. O. Alvares et al., *An algorithm to identify avoidance behavior in moving object trajectories*, Journal of the Brazilian Comp. Society, 17 (2011), pp. 193–203.
[4] W. Application, *Waze map safety re-route*, 2017. http://www.cbsnews.com/waze-map-app-suicide-straight-safety-reroute-high-crime-neighborhoods/.
[5] R. Arnott and K. Small, *The economics of traffic congestion*, American scientist, (1994), pp. 446–455.
[6] C. Chen et al., *Real-time detection of anomalous taxi trajectories from gps traces*, in Int. Conf. on Mobile and Ubiquitous Systems, Springer, 2011, pp. 63–74.
[7] C. Cheng et al., *Where you like to go next: Successive point-of-interest recommendation.*, in IJCAI, vol. 13, 2013, pp. 2605–2611.
[8] M. Egenhofer, *A model for detailed binary topological relationships*, Geomatica, 47 (1993), pp. 261–273.
[9] Y. Fu et al., *Real estate ranking via mixed land-use latent models*, in In Proc. of 21th SIGKDD, ACM, 2015, pp. 299–308.
[10] Y. Ge et al., *Top-eye: Top-k evolving trajectory outlier detection*, in Proc. of 19th ACM Int. Conf. of CIKM, ACM, 2010, pp. 1733–1736.
[11] F. Giannotti et al., *Trajectory pattern mining*, in In Proc. of 13th SIGKDD, ACM, 2007, pp. 330–339.
[12] M. Haklay and P. Weber, *Openstreetmap*, Pervasive Computing, 7 (2008), pp. 12–18.
[13] P. E. Hart et al., *A formal basis for the heuristic determination of minimum cost paths*, Trans. on Sys. Science and Cybernetics, 4 (1968), pp. 100–107.
[14] P. Karich and S. Schröder, *Graphhopper*, http://www. graphhopper. com, (2014).
[15] S. Leman-Langlois, *The local impact of police video-surveillance on the social construction of security*, Technocrime: Technology, Crime and Social Control, (2008), pp. 27–45.
[16] F. Lettich et al., *Detecting avoidance behaviors between moving object trajectories*, Data & Knowledge Engineering, 102 (2016), pp. 22–41.
[17] S. T. Leutenegger et al., *Str: A simple and efficient algorithm for r-tree packing*, in Proc. 13th Int. Conf. on Data Engineering, IEEE, 1997, pp. 497–506.
[18] P. Newson and J. Krumm, *Hidden markov map matching through noise and sparseness*, in Proc. of the 17th SIGSPATIAL Conf., ACM, 2009, pp. 336–343.
[19] B. E. Notar, *10 off limits and out of bounds taxi driver perceptions of dangerous people and places in kunming, china*, Rethinking Global Urbanism, (2012), pp. 190–207.
[20] P. Wang et al., *Human mobility synchronization and trip purpose detection with mixture of hawkes processes*, in In Proc. of 23th SIGKDD, ACM, 2017, pp. 495–503.
[21] D. B. West et al., *Introduction to graph theory*, vol. 2, Prentice Hall, 2001.
[22] C. Xu et al., *Identifying travel mode from gps trajectories through fuzzy pattern recognition*, in Int. Conf. on Fuzzy Systems and Knowledge Discovery, vol. 2, IEEE, 2010, pp. 889–893.
[23] H. Yoon et al., *Smart itinerary recommendation based on user-generated gps trajectories*, in International Conference on Ubiquitous Intelligence and Computing, Springer, 2010, pp. 19–34.
[24] J. Yuan et al., *T-drive: driving directions based on taxi trajectories*, in Proc. of the 18th SIGSPATIAL, ACM, 2010, pp. 99–108.
[25] J. Yuan et al., *Discovering regions of different functions in a city using human mobility and pois*, in In Proc. of 18th SIGKDD, ACM, 2012, pp. 186–194.
[26] Q. Yuan et al., *Time-aware point-of-interest recommendation*, in Proc. of 36th Int. SIGIR Conf. on Research and Development in Information Retrieval, ACM, 2013, pp. 363–372.
[27] D. Zhang et al., *ibat: detecting anomalous taxi trajectories from gps traces*, in Proc. of 13th Int. Conf. on Ubiquitous Comp., ACM, 2011, pp. 99–108.
[28] L. Zhang, S. Dalyot, D. Eggert, and M. Sester, *Multi-stage approach to travel-mode segmentation and classification of gps traces*, Int. Arch. of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 38 (2011), pp. 87–93.
[29] Y. Zheng et al., *Understanding mobility based on gps data*, in Proc. of 10th Int. Conf. on Ubiquitous Comp., ACM, 2008, pp. 312–321.
[30] Y. Zheng et al., *Mining interesting locations and travel sequences from gps trajectories*, in Proc. of 18th Int. Conf. on WWW, ACM, 2009, pp. 791–800.
[31] Y. Zheng et al., *Understanding transportation modes based on gps data for web applications*, Transactions on the Web (TWEB), 4 (2010), p. 1.
[32] Y. Zheng and X. Zhou, *Computing with spatial trajectories*, Springer Science & Business Media, 2011.