

2011

Privacy-Preserving Assessment of Location Data Trustworthiness

Chenyun Dai

Fang-Yu Rao

Gabriel Ghinita

Elisa Bertino

Purdue University, bertino@cs.purdue.edu

Follow this and additional works at: <http://docs.lib.purdue.edu/ccpubs>

 Part of the [Engineering Commons](#), [Life Sciences Commons](#), [Medicine and Health Sciences Commons](#), and the [Physical Sciences and Mathematics Commons](#)

Dai, Chenyun; Rao, Fang-Yu; Ghinita, Gabriel; and Bertino, Elisa, "Privacy-Preserving Assessment of Location Data Trustworthiness" (2011). *Cyber Center Publications*. Paper 394.
<http://docs.lib.purdue.edu/ccpubs/394>

This document has been made available through Purdue e-Pubs, a service of the Purdue University Libraries. Please contact epubs@purdue.edu for additional information.

Privacy-Preserving Assessment of Location Data Trustworthiness

Chenyun Dai, Fang-Yu Rao, Gabriel Ghinita and Elisa Bertino

Dept. of Computer Science, Purdue University

email: {daic, raof, gghinita, bertino}@purdue.edu

ABSTRACT

Assessing the trustworthiness of location data corresponding to individuals is essential in several applications, such as forensic science and epidemic control. To obtain accurate and trustworthy location data, analysts must often gather and correlate information from several independent sources, e.g., physical observation, witness testimony, surveillance footage, etc. However, such information may be fraudulent, its accuracy may be low, and its volume may be insufficient to ensure highly trustworthy data. On the other hand, recent advancements in mobile computing and positioning systems, e.g., GPS-enabled cell phones, highway sensors, etc., bring new and effective technological means to track the location of an individual. Nevertheless, collection and sharing of such data must be done in ways that do not violate an individual's right to personal privacy.

Previous research efforts acknowledged the importance of assessing location data trustworthiness, but they assume that data is available to the analyst in direct, unperturbed form. However, such an assumption is not realistic, due to the fact that repositories of personal location data must conform to privacy regulations. In this paper, we study the challenging problem of refining trustworthiness of location data with the help of large repositories of anonymized information. We show how two important trustworthiness evaluation techniques, namely *common pattern analysis* and *conflict/support analysis*, can benefit from the use of anonymized location data. We have implemented a prototype of the proposed privacy-preserving trustworthiness evaluation techniques, and the experimental results demonstrate that using anonymized data can significantly help in improving the accuracy of location trustworthiness assessment.

Categories and Subject Descriptors

H.2.8 [Database Applications]: Spatial databases and GIS

General Terms

Security, Experimentation

Keywords

Data Trustworthiness, Location Data, Privacy

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ACM SIGSPATIAL GIS '11, November 1-4, 2011, Chicago, IL, USA
Copyright 2011 ACM 978-1-4503-1031-4/11/11 ...\$10.00.

1. INTRODUCTION

For law-enforcement and homeland security agencies, it is often very important to verify the location information of certain individuals, in order to perform forensic investigations [19] or manage natural disasters, control epidemics [1], etc. The recent advances in mobile computing and positioning systems, e.g., GPS-enabled cell phones, provide the technological means to find out the location of an individual. However, even with extremely rich location information provided by those modern devices, it is sometimes difficult to find one's true location due to data loss, malicious data modification, or device limitations. For instance, a crime suspect is highly likely to avoid being associated with a crime scene, therefore s/he may turn off his or her phone. A potential disease carrier may lie about the locations s/he visited because of the fear of immediate isolation and infection control procedures [8]. Furthermore, mobile devices may be disabled for a period of time because of lack of signal or power.

Therefore, it may be necessary to corroborate information originating at multiple sources in order to determine with high trustworthiness the actual location of an individual at a given time. There are many repositories of data that could provide valuable location information, such as surveillance cameras, highway traffic monitoring sensors, etc. However, sharing of such information is governed by certain privacy laws, that ensure that the personal details of innocent individuals are being protected. For example, law enforcement agencies can obtain exact trajectories of an individual only if they have a valid search warrant [2]. In many cases, at the early stages of an investigation when there is a lack of evidence, it is difficult to obtain such a search warrant. In addition, to preserve individual privacy, some data gathering entities, e.g., traffic management systems, highway sensors, collect only anonymous data that is further transformed before being sent to the database server [3]. As original data are permanently modified, it is not possible to link a particular device with a user. In such a case, the location verification procedure is not straightforward, and an in-depth analysis step of the anonymized data must be performed to deal with the inherent information loss and to yield useful results.

Consider the example in Figure 1, where the police department is investigating a crime that occurred at location z , and there are three suspects traveling on the highway leading from x to y through z . The investigators were able to identify three trajectories (blue lines) that correspond to these three suspects, namely Alice, Bob and Carol. According to the witnesses, Alice, Bob and Carol were together at location x at 1pm and at location y at 3pm, respectively. But only Alice and Carol were observed by another witness at location z at 2pm. The location of Bob at 2pm was self-reported as being at the midpoint m of one of the other roads from x to y on the map. Witnesses have observed Alice and Carol's car, but not Bob's.

To further the investigation, the police department consults a traffic management system (TMS) which monitors the highway using sensors. Since the location information recorded by the TMS is available only in anonymized form, the police department receives a set of anonymized trajectories (red lines), and their associated enclosing cylinder with a diameter of δ (we provide formal details about the anonymization model used in Section 2). If the anonymized trajectories are in fact the anonymized versions of the trajectories of Alice, Bob and Carol, then the police can conclude that Bob’s trajectory should be one of the three anonymized trajectories, since nobody else was present in the cylinder area. Although it may not be possible to tell which one of the anonymized trajectories exactly corresponds to Bob, it is highly likely that Bob was in location z at 2pm, instead of the reported location m .

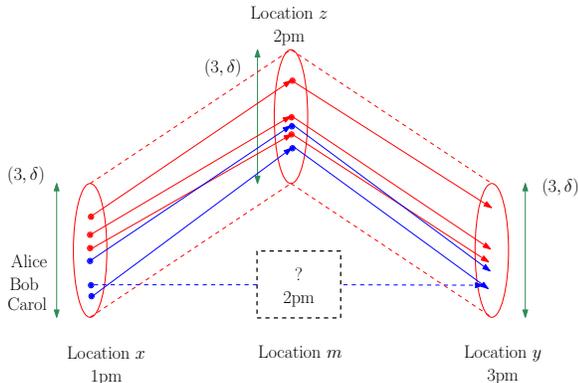


Figure 1: Trustworthiness Assessment with Anonymized Data

The example above illustrates how a repository of anonymized location data can be used to assess the trustworthiness of data that is already available in exact form, but which may not always be truthful. Such assessment can be performed through two distinct types of analyses: *common pattern* analysis and *conflict/support* analysis. Specifically, if the majority of movements between a source and destination location follow a certain pattern, i.e., from x to z and then to y as is the case for all three anonymized trajectories, then reported trajectories that deviate from this pattern may be less trustworthy. In this case, the trajectory reported by Bob does not appear trustworthy due to its dissimilarity with all other movements. Note that, since anonymized data is typically available in large volumes, as large populations of users are often observed, the reliability of such movement patterns can be properly estimated, even if the anonymized data is not entirely accurate. In other words, *aggregate-level* information is preserved, despite the inherent information loss about individual trajectories. Hence, anonymized data from a large number of observations is very suitable for common pattern analysis.

Furthermore, given the anonymized information, we can try to link the data with the exact data available to the investigators. In other words, it is possible to link the *local* dataset of exact information with the *remote* dataset of anonymized information. Since there are three anonymized trajectories, and they all begin at x and end at y , and three exact observations at these respective endpoints corresponding to Alice, Bob and Carol’s trajectories, then since there is no other individual that could have generated the reading of the third observation at location z , it can be inferred that Bob was indeed at z , and not at the self-reported location m . This way, even though there are no identifiers in the anonymized data, through a careful mapping of available trajectories it is possible to perform a conflict/support analysis to evaluate the trustworthiness of a local reading. In other words, it is possible to determine whether the

local information is supported by the remote anonymized data, or contradicted.

This paper contributes mechanisms to perform common pattern and conflict/support analyses for location data trustworthiness assessment in the presence of anonymized data. In practice, the amount of data available to investigators in a single case is relatively small in size, and large repositories of anonymized data can significantly improve the process of assessing data trustworthiness, despite the inaccuracy incurred in the anonymization process. The rest of the paper is organized as follows: Section 2 introduces the data and privacy model adopted in this paper, as well as other fundamental concepts and definitions. Section 3 presents the algorithms for computation of trajectory trust scores. We report experimental results on real datasets in Section 4. Section 5 reviews related work, whereas Section 6 concludes the paper and outlines future research directions.

2. SYSTEM ARCHITECTURE AND PRIVACY MODEL

In this section, we introduce our system and privacy models. Figure 2 illustrates our system architecture. Our system has one local dataset and several remote datasets. Our goal is to refine the trustworthiness of the local dataset by comparing our local dataset with remote datasets. As an example, in Figure 2, the law enforcement agency queries two remote datasets, namely a cell phone company and a traffic management system, for additional information. The two remote parties then send their anonymized datasets to the law enforcement agency, which later on performs trustworthiness score computation based on both local datasets and anonymized datasets. Our architecture can adopt any generalization-based anonymization models.

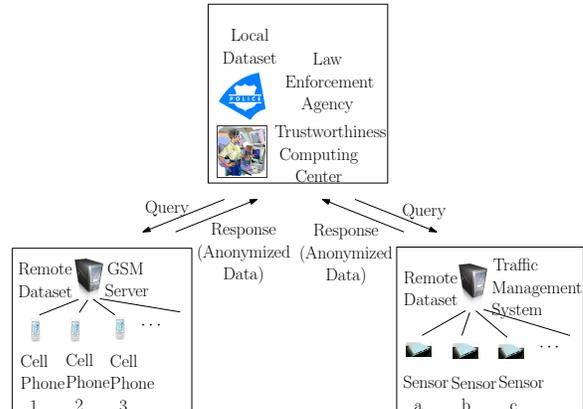


Figure 2: System Architecture

Following [10], a trajectory is a polyline in three-dimensional space represented as a sequence of spatio-temporal points, and each point is associated with a unique identifier.

DEFINITION 1. Trajectory: A position \mathcal{P} is defined as a tuple (oid, x, y, t) , where oid is a unique identifier of an object and (x, y, t) is a spatio-temporal point. A trajectory \mathcal{T} is a sequence of time-stamped positions $\mathcal{P}_0 = (oid, x_0, y_0, t_0), \dots, \mathcal{P}_k = (oid, x_k, y_k, t_k)$ where $\forall_{0 \leq i < k} (t_i < t_{i+1})$. We denote the position \mathcal{P}_i of \mathcal{T} by $\mathcal{T}.\mathcal{P}_i$, and we call \mathcal{P}_{i+1} the successor of \mathcal{P}_i . \square

DEFINITION 2. Movement: A movement \mathcal{M} is a pair of position \mathcal{P}_i and its successor \mathcal{P}_{i+1} within a trajectory.

DEFINITION 3. Group of Movements: Let \mathcal{M} be a movement, and let $\mathcal{M.P}_s$ and $\mathcal{M.P}_e$ denote its start and end positions. A group G of movements from cluster C_x of positions to another cluster C_y is a set of movements such that for each movement $\mathcal{M} \in G$, $D(\mathcal{M.P}_s, \mu(C_x)) \leq \theta$ and $D(\mathcal{M.P}_e, \mu(C_y)) \leq \theta$ where $D(\cdot, \cdot)$ is the two-dimensional Euclidean distance of the two positions and $\mu(C_x)$, $\mu(C_y)$ are the centroids of the cluster C_x and C_y respectively.

DEFINITION 4. Pattern: A pattern PT is a set of movements from one cluster C_x of positions to another cluster C_y such that for each movement $\mathcal{M} \in PT$, (s, e) resides in a bounding circle of radius ϵ on the plane of start time and end time where s is the start time of \mathcal{M} and e is the end time of \mathcal{M} .

DEFINITION 5. Trust Score: The trust score, denoted as $S(\cdot)$, ranging from 0 to 1, indicates the probability of location information being correct. \square

Location information can be either a position \mathcal{P} , a trajectory \mathcal{T} , a movement \mathcal{M} or a pattern PT . Higher scores indicate higher trustworthiness. The use of *trust score* can vary depending on different applications. In the forensic analysis example, police officers can rank the suspects based on the trust scores of those suspects being at the crime scene, and then start the investigation from the suspect with the highest score. They can also rank the suspects based on the trust scores of those suspects not being at the crime scene and start the investigation from the suspect with the lowest score. Trust score is not the exact probability, however, it can be used as an indication of probability.

2.1 Privacy Model

The *Never Walk Alone (NWA)* [4] system for anonymizing trajectories proposed the (k, δ) -anonymity model, where δ represents the imprecision in the positioning systems used. The basic idea is to move some positions along each trajectory so that all trajectories in the anonymized dataset satisfy the (k, δ) anonymity requirement. In our privacy model, we adopt this concept. We briefly introduce related definitions in the following.

DEFINITION 6. Co-localization: Two trajectories $\mathcal{T}_1, \mathcal{T}_2$ defined in $[t_1, t_n]$ are said to be co-localized w.r.t. δ , iff for each position (o_1, x_1, y_1, t) in \mathcal{T}_1 and (o_2, x_2, y_2, t) in \mathcal{T}_2 with $t \in [t_1, t_n]$, it holds that $Dist((x_1, y_1), (x_2, y_2)) \leq \delta$, where $Dist$ is the Euclidean distance. We also write $Coloc_\delta(\mathcal{T}_1, \mathcal{T}_2)$ if \mathcal{T}_1 and \mathcal{T}_2 are co-localized w.r.t. δ .

DEFINITION 7. k -Anonymity Set of Trajectories: Given a position uncertainty threshold δ and an anonymity threshold k , a set S of trajectories is a (k, δ) -anonymity set iff $|S| \geq k$ and $\forall \mathcal{T}_i, \mathcal{T}_j \in S, Coloc_\delta(\mathcal{T}_i, \mathcal{T}_j)$.

Figure 3 is an example of $(2, \delta)$ -anonymity set formed by 2 co-localized trajectories.

Note that in Definition 6, identifiers o_1 and o_2 are only used to distinguish trajectories. In other words, it can be a random number for each trajectory and cannot be used to link to a specific individual. Different remote providers may use different identifiers for trajectories belonging to the same individual. In this model, a remote provider anonymizes its original dataset to fulfill the (k, δ) -anonymity requirement and then sends it to the local provider.

Here we also briefly describe how the anonymization of trajectories is done in [4]. First, all the trajectories will be partitioned into different equivalence classes with respect to their time spans.

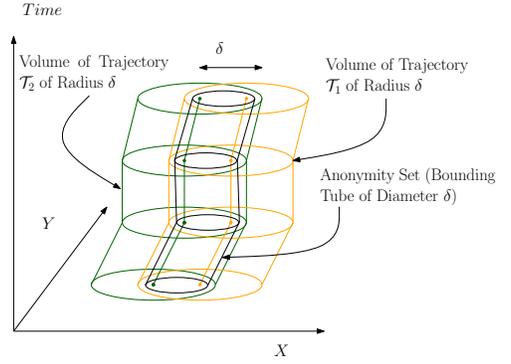


Figure 3: An example of a $(2, \delta)$ -set formed by 2 co-localized trajectories, their respective uncertainty volumes, and the central cylinder of diameter δ containing both trajectories.

Two trajectories are considered to be in the same equivalence class if they have same start time and end time. Second, for each of these equivalence classes produced, a clustering algorithm does the clustering on the trajectories within this equivalence class and then produces a set of clusters of trajectories such that any two trajectories within a cluster are close to each other. Finally, for each cluster mentioned above, some positions of points within the cluster will be modified so that any two of the trajectories are co-localized to each other.

Notice in the Definition 6, two trajectories to be co-localized must be defined over the same time interval. Although it's not usual for two trajectories starting and ending at the exact same time, this problem can be dealt with by allowing small time gaps, or by choosing coarser time samplings.

3. COMPUTING TRUSTWORTHINESS OF TRAJECTORIES

In this section, we present our algorithms for computing the trustworthiness of positions. The trust scores are computed by taking into account two types of analysis: (1) *common pattern analysis* and (2) *conflict/support analysis*. The *common pattern analysis* extracts the similarities between movements within the local and remote datasets, while *conflict/support analysis* determines the conflicts and supports between trajectories from the local dataset and the remote dataset.

3.1 Common Pattern Analysis

The goal of common pattern analysis is to extract a set of movement patterns from both the local and the remote data sources. Such patterns will be later used to measure the trustworthiness of a given trajectory. Algorithm 1 works as follows: given a set of local trajectories and a set of remote trajectories, it will first produce a set of groups of local movements by calling the procedure `CLUSTERINGLOCALPOSITIONS`, then refine those clusters by taking into account anonymized trajectories from a remote source — in procedure `CLUSTERINGREMOTEPOSITIONS`. Then for each group of movements, it will further partition those movements into different patterns according to their corresponding start times and end times by the call to the procedure `GROUPINGPATTERNS`.

We emphasize that, the reason we would like to take remote trajectories into consideration is that it's possible those remote trajectories contain some patterns we do not have locally. However, because of the uncertainty of those remote trajectories, we do not want the quality of our local data to be deteriorated by including all

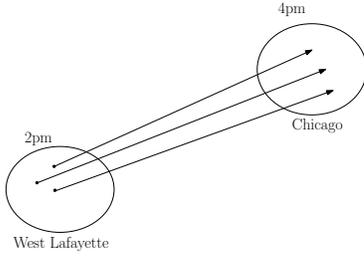


Figure 4: An Example of a Good Clustering

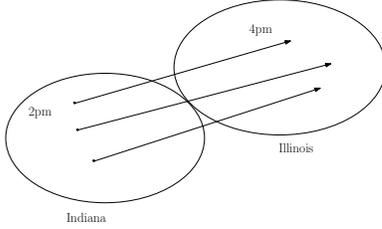


Figure 5: An Example of a Bad Clustering

of those remote trajectories. Thus, in some cases, we will choose to include a remote trajectory while in some other cases, we will choose to discard a remote trajectory.

Note that in CLUSTERINGLOCALPOSITIONS, we need to choose a threshold ρ , which is the upper bound for the approximate bounding circles of clusters of positions. The choice of ρ requires some careful fine-tuning, since on the one hand, we do not want this ρ to be too large because in this case, the radii of those approximate bounding circles might be so large that those circles intersect each other, which makes it difficult for us to identify the groups of movements later and also make the data quality worse. For instance, in Figure 4 (movements from West Lafayette to Chicago), we can see the pattern in this group of movements is more accurate compared to the pattern in Figure 5 (movements from Indiana to Illinois). In addition, we would also like to make room for some other possible locations that we don't have locally which are provided by the remote party.

But on the other hand, we don't want ρ to be too small since it might induce too many small clusters of positions and small groups of movements in CLUSTERINGLOCALPOSITIONS and GROUPINGMOVEMENTS, which will make the patterns less obvious. Ideally, those remote positions corresponding to the objects that we have will reside in the approximate bounding circles for local positions with radii upper bounded by ρ while those positions corresponding to other objects that we don't have locally can be taken in without creating a bounding circle intersecting with other existing ones. In our experiments (Section 4), we choose ρ to be 5% of the diagonal of the minimum bounding rectangle of our dataset.

Notice in the procedure CLUSTERINGREMOTEPOSITIONS we need to compute the radius of the approximate bounding circle of a cluster $C \cup \{\mathcal{P}\}$ where \mathcal{P} is a remote position. This can be computed using the procedure RADIUSIFADDED. In addition, two approximate bounding circles K_1, K_2 (centered at c_1 and c_2 respectively) intersect if $D(c_1, c_2) - (r_1 + r_2) < 0$ where r_1 and r_2 are their corresponding radii. Also, the radius of a bounding circle containing only a remote position \mathcal{P} is set to ρ , the threshold we used in CLUSTERINGLOCALPOSITIONS. Last thing to note is the relaxation factor σ in line 6 of CLUSTERINGREMOTEPOSITIONS. We found in experiments that if there is not enough data in the local

Algorithm 1: COMMONPATTERNANALYSIS

Input: A set Ω_L of local trajectories, a set Ω_R of remote trajectories preprocessed by NWA, a radius threshold ρ , a relaxation factor σ , and a time threshold ϵ .

Output: A group of patterns with their corresponding scores.

- 1 $\Pi_L \leftarrow$ the set of positions extracted from Ω_L ;
 - 2 $\Pi_R \leftarrow$ the set of positions extracted from Ω_R ;
 - 3 $\mathcal{C} \leftarrow$ CLUSTERINGLOCALPOSITIONS(Π_L, ρ);
 - 4 $\mathcal{C}' \leftarrow$ CLUSTERINGREMOTEPOSITIONS($\mathcal{C}, \Pi_R, \sigma$);
 - 5 $\mathcal{G} \leftarrow$ GROUPINGMOVEMENTS(\mathcal{C}');
 - 6 **foreach** group of movement $K \in \mathcal{G}$ **do**
 - 7 $\mathcal{PT} \leftarrow$ GROUPINGPATTERNS(K, ϵ);
 - 8 **end**
 - 9 **foreach** pattern $PT \in \mathcal{PT}$ **do**
 - 10 assign a score to PT ;
 - 11 **end**
 - 12 **return** $\{PT_1, \dots, PT_n\}$
-

Procedure CLUSTERINGLOCALPOSITIONS

Input: A set S of local positions, a threshold ρ .

Output: A set $\mathcal{C} = \{C_1, \dots, C_m\}$ of clusters of positions.

- 1 **foreach** position \mathcal{P} from S **do**
 - 2 **if** there is no cluster C such that $D(\mathcal{P}, \mu(C)) \leq \rho$ **then**
 - 3 make a new cluster C_i ;
 - 4 include \mathcal{P} into C_i ;
 - 5 **else**
 - 6 find the cluster C closest to \mathcal{P} ;
 - 7 include \mathcal{P} into C ;
 - 8 **end**
 - 9 **end**
 - 10 **return** $\{C_1, \dots, C_m\}$;
-

dataset, then by carefully setting σ to a value a little greater than 1, more remote positions and movements could be included, which can also help when computing the trustworthiness of movements in local dataset.

Given a position \mathcal{P} , we can compute the trust score for \mathcal{P} by computing the average score of the movement(s) in which \mathcal{P} is the start position or the end position. To compute a score for a given movement \mathcal{M} from a location x (at time s) to a location y (at time e), we need to first find out the source cluster C_x of positions that contains x and the destination cluster C_y of positions that contains y . If there exist such C_x and C_y , then according to these two clusters of positions, we are able to construct the group of movements from C_x to C_y and thus the corresponding pairs of start time and end time. The score of \mathcal{M} may be computed as follows:

$$S(\mathcal{M}) = \begin{cases} S(PT) & \text{if } D((s, e), \mu(PT)) < \epsilon \\ \frac{\epsilon}{D((s, e), \mu(PT))} \cdot S(PT) & \text{otherwise} \end{cases}$$

where PT is the closest pattern to (s, e) on the plane of start time and end time, and ϵ is the parameter used in the procedure GROUPINGPATTERNS. From above we know if (s, e) falls in the bounding circle of PT , the score of \mathcal{M} would be equal to those movements in PT while the score of \mathcal{M} is inversely proportional to its distance from the center of PT when (s, e) falls outside the bounding circle of PT .

However, if either one or both of C_x and C_y does not exist, then we simply assign 0 to \mathcal{M} since we do not have any pattern that is similar to the movement \mathcal{M} from x to y . Specifically, we consider that C_x does not exist if there is no cluster of positions C such that

Procedure CLUSTERINGREMOTEPOSITIONS

Input: A set $\mathcal{C}_L = \{C_1, \dots, C_m\}$ of clusters of local positions, a set T of remote positions, a relaxation factor σ .

Output: A set $\mathcal{C}' = \{C_1, \dots, C_{m'}\}$ of clusters of positions.

```
1 foreach cluster  $C_i \in \mathcal{C}_L$  do
2   |  $r_i \leftarrow \max_{\mathcal{P} \in C_i} D(\mathcal{P}, \mu(C_i));$ 
3 end
4  $r_{max} \leftarrow \max\{r_1, \dots, r_m\};$ 
5 foreach position  $\mathcal{P} \in T$  do
6   | if there is no cluster  $C$  such that the radius of the
   | approximate bounding circle of  $C \cup \{\mathcal{P}\} \leq \sigma \cdot r_{max}$  then
7     | if the approximate bounding circle of  $\{\mathcal{P}\}$  intersects
   | any other approximate bounding circle then
8       | discard  $\mathcal{P}$ ;
9     | else
10      | make a new cluster  $C'$  and include  $\mathcal{P}$  into  $C'$ ;
11      | end
12    | else
13      | find the cluster  $C$  closest to  $\mathcal{P}$ ;
14      | include  $\mathcal{P}$  into  $C$ ;
15    | end
16 end
17 return  $\{C_1, \dots, C_{m'}\};$ 
```

Procedure RADIUSADDED

Input: A cluster C of positions, a remote position P , and ρ , the threshold used in CLUSTERINGLOCALPOSITIONS.

Output: The radius r of the approximate bounding circle of $C \cup \{P\}$.

```
1 find a local position  $\mathcal{Q}_L \in C$  among all the local positions in
   $C$  such that
   $D(\mathcal{Q}_L, \mu(C \cup \{P\})) = \max_{\mathcal{P}_L \in C \cup \{P\}} D(\mathcal{P}_L, \mu(C \cup \{P\}));$ 
2 find a remote position  $\mathcal{Q}_R \in C$  among all the remote positions
  in  $C$  such that
   $D(\mathcal{Q}_R, \mu(C \cup \{P\})) = \max_{\mathcal{P}_R \in C \cup \{P\}} D(\mathcal{P}_R, \mu(C \cup \{P\}));$ 
3  $r \leftarrow \max\{D(\mathcal{Q}_L, \mu(C \cup \{P\})), D(\mathcal{Q}_R, \mu(C \cup \{P\})) + \rho\};$ 
4 return  $r;$ 
```

the distance between location x and the centroid of C is less than or equal to r_{max} , the maximum radius of those clusters of positions we got after the call to the procedure CLUSTERINGLOCALPOSITIONS.

Next we explain how to assign a score to each pattern produced in the procedure GROUPINGPATTERNS. Let C_{PTs} be the source cluster of positions of the pattern PT and C_{PTt} the destination cluster of positions of the pattern PT . Let K be the group of movements from which PT is built (recall that $PT \subseteq K$). The score of a pattern PT can be computed as follows:

$$S(PT) = \begin{cases} \min\{|PT|/|K'|, 1\} & \text{if } |K| > \lambda \\ \min\left\{\frac{|PT|}{\sum_{i=1}^m |G'_i| + \sum_{j=1}^n |H'_j| + |K'|}, 1\right\} & \text{if } |K| \in [\beta, \lambda] \\ |PT|/|T| & \text{if } |K| < \beta \end{cases}$$

where $G'_i \subseteq G_i$ is the set of all local movements in G_i , a group of movements (containing both local and remote ones) not equal to K that starts from the cluster C_{PTs} , $H'_j \subseteq H_j$ is the set of all local movements in H_j , a group of movements not equal to K that ends with the cluster C_{PTt} . Similarly, $K' \subseteq K$ is the set

Procedure GROUPINGMOVEMENTS

Input: A set $\mathcal{C} = \{C_1, \dots, C_m\}$ of clusters of positions.

Output: A set $\mathcal{G} = \{G_1, \dots, G_n\}$ of groups of movements.

```
1 foreach cluster  $C_i \in \mathcal{C}$  do
2   | foreach  $\mathcal{P}_x \in C_i$  do
3     | if there does not exist  $C_j$  such that  $\text{succ}(\mathcal{P}_x) \in C_j$  or
   |  $\mathcal{P}_x$  and  $\text{succ}(\mathcal{P}_x)$  are both in  $C_i$  then
4       |  $C_i \leftarrow C_i \setminus \{\mathcal{P}_x\};$ 
5       | end
6     | end
7   | foreach unmarked position  $\mathcal{P}_\ell$  in  $C_i$  do
8     | create a new group  $G_k$ ;
9     | add the movement  $(\mathcal{P}_\ell, \text{succ}(\mathcal{P}_\ell))$  to  $G_k$ ;
10    | mark  $\mathcal{P}_\ell$ ;
11    | foreach each unmarked position  $\mathcal{P}_y$  in  $C_i$  do
12      | if both  $\text{succ}(\mathcal{P}_\ell)$  and  $\text{succ}(\mathcal{P}_y)$  are in the same
   | cluster then
13        | add the movement  $(\mathcal{P}_y, \text{succ}(\mathcal{P}_y))$  to  $G_k$ ;
14        | mark  $\mathcal{P}_y$ ;
15      | end
16    | end
17  | end
18 end
19 return  $\{G_1, \dots, G_n\};$ 
```

Procedure GROUPINGPATTERNS

Input: A set T of the pairs of start time and end time for a group of movements (with similar sources and destinations), ϵ .

Output: A set $\mathcal{PT} = \{PT_1, \dots, PT_n\}$ of patterns.

```
1 foreach pair  $(s, e)$  of start time and end time in  $T$  do
2   | if there is no cluster  $C$  such that  $D((s, e), \mu(C)) \leq \epsilon$  then
3     | make a new cluster  $C_i$  and include  $(s, e)$  into  $C_i$ ;
4     | else
5       | find the cluster  $C$  closest to  $(s, e)$ ;
6       | include  $(s, e)$  into  $C$ ;
7     | end
8 end
9 return  $\{PT_1, \dots, PT_n\};$ 
```

of all local movements in K . T is the union of all the groups of movements (both local and remote ones), $\beta = \alpha|T|$, $\alpha \in [0, 1]$, $\lambda = \gamma|T|$ and $\gamma \in [0, 1]$. Note also that if all the G'_i , H'_j , and K' are empty, then we will give this movement a 0 score since we do not have any local evidence for this movement. The reason we define the score function this way is that we'd like to categorize all the groups into 3 classes: (1) common, (2) not common but neither rare, and (3) rare. If a group K from cluster C_x to cluster C_y is really large (above the threshold $\gamma|T|$), then the scores of patterns in K should not be affected because of adding other groups that also start from cluster C_x or end with cluster C_y . If K is not that large ($\alpha|T| < |K| < \gamma|T|$), then we should take into consideration those groups that also start from cluster C_x or end with cluster C_y when calculating the scores for patterns in $|K|$.

However, we also have to consider the situation in which there are very few movements within a group ($|K| < \alpha|T|$). For example, consider the case in which $PT = \{\mathcal{M}\}$ where \mathcal{M} is a movement from location x (2pm) to location y (3pm). If $C_{PTs} = \{x\}$, $C_{PTt} = \{y\}$ and $K = \{\mathcal{M}\}$, then we can see this pattern is pretty

rare compared to other patterns and thus we would like to give this kind of patterns a lower score.

Whether a pattern PT is rare (or common) should be decided by the user. This can be done by adjusting the parameter α in the scoring function. For instance, if α is set to 0.001, then a pattern PT would be considered rare if the cardinality of the group of movements from which PT is built is less than 0.1 percent of the cardinality of T , the union of all the groups of movements.

Another reason why we have this score function is that we do not want the score of a local movement to be decreased by including those movements from another party unless this local movement is “really” rare with respect to $|T|$, the cardinality of the union of all the groups of movements. That’s also why in the procedure CLUSTERINGREMOTEPOSITIONS, sometimes we still need to create a new cluster containing a remote position. In this way, a movement from a remote position in another cluster to this newly constructed cluster could be included in T .

From the score function above, it seems a movement will be penalized because of being rare and thus will be given a low score. But if this movement is truthful, it will be given a higher score in the conflict and support analysis later so a truthful movement will not be treated as a fake one. Also, the reason why we do not assign a zero score to PT in the third case is that we would like to differentiate a movement \mathcal{M}_1 that does not match any movement in our database (i.e. its corresponding C_x or C_y does not exist or there is no movement from C_x to C_y directly) from a movement \mathcal{M}_2 that is similar to some movements in our database when calculating the trust scores for them. If a movement \mathcal{M} does not match any movement in our database, it will be assigned a zero score while if \mathcal{M} matches some rare pattern in our dataset, it will be assigned some non-zero score although it’s still much lower when compared to the scores of other common patterns.

Figure 6 illustrates how we assign a score to a pattern PT . Suppose in this case, β is set to 2 and λ is set to 10 (so K is not a “common” group) and thus we will use the second part of the score function to compute the score. Also we assume right now those groups contain only “local” movements. Therefore, $|G'_1| = 2$, $|G'_2| = 2$, $|G'_3| = 2$, $|H'_1| = 2$, $|H'_2| = 2$, $|K'| = 4$, and $|PT| = 2$. Therefore, $S(PT) = 2/((2 + 2 + 2) + (2 + 2) + 4) = 2/14$ and every movement in PT has the same score. If later we include a remote movement \mathcal{M}_R from C_{PT_s} (2pm) to C_{PT_t} (4pm), then this movement will be added to K and also to the pattern PT . The score of this pattern will be raised to $3/((2 + 2 + 2) + (2 + 2) + 4) = 3/14$ since $|PT|$ is equal to 3 now.

3.2 Conflict/Support Analysis

In this section, we propose an approach for computing the trust scores of a position \mathcal{P} in the local dataset based on whether it conflicts or is supported by a position in the remote dataset. Our approach takes two input datasets: 1) local dataset which contains exact trajectories; 2) remote dataset which contains anonymized data. Our goal is to use the remote data to refine the trustworthiness of the local ones based on *conflict/support* information. Before we propose algorithms, we first define the concept of support and conflict. Note that in our privacy model, it is impossible to have identifiers for positions and trajectories. Without identifiers, we are not able to link two trajectories corresponding to the same individual precisely. The only thing that can be done is to estimate the likelihood of a local trajectory corresponding to a remote one and then map the local trajectory to a remote one based on this likelihood. After the mapping, we treat the two mapped trajectories as if they have the same identifier. We then adjust the trust scores of each position of the local trajectory based on whether it conflicts

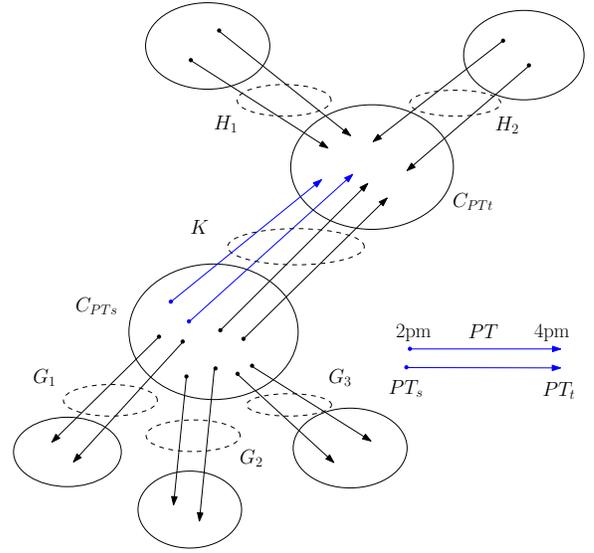


Figure 6: An Example of How to Compute the Score for PT

the remote trajectory or is supported by the remote one.

We introduce the definition of conflict and support as follows:

DEFINITION 8. Support and Conflict Let \mathcal{T} and \mathcal{T}' be two different trajectories of the same object. Let $\mathcal{P}_a = (oid, x_a, y_a, t_a)$ be a position in \mathcal{T} , $\mathcal{P}_b = (oid, x_b, y_b, t_b)$ a position in \mathcal{T}' such that $t_b = t_a$. Given a threshold ψ , if $D(\mathcal{P}_a, \mathcal{P}_b) \leq \psi$, we say that \mathcal{P}_a and \mathcal{P}_b support each other; otherwise (i.e., $D(\mathcal{P}_a, \mathcal{P}_b) > \psi$), we say that \mathcal{P}_a conflicts with \mathcal{P}_b . \square

Here, ψ is a user specified threshold that determines whether two positions support or conflict with each other. Determining a good ψ value is a very difficult work. If ψ is set too small, most positions in local dataset will have counterparts in remote dataset that conflict with them. However, if ψ is too big, most positions in the local dataset will be supported by one in remote dataset. We will show in the experiments how different ψ values affect the experimental results. As mentioned above, since we cannot be sure which remote trajectory corresponds to which local one, the only thing we can do is to make a best-effort assignment based on the similarity of each local/remote trajectory pair. To do that, we first define the distance between trajectories.

DEFINITION 9. Distance between Two Trajectories Let \mathcal{T} be a local trajectory and \mathcal{T}' be a remote trajectory defined in the same time span $[t_1, t_n]$. We define the average distance between \mathcal{T} and \mathcal{T}' as $(\sum_{i=1}^n D(\mathcal{P}_i, \mathcal{P}'_i))/n$, denoted as $AvgD(\mathcal{T}, \mathcal{T}')$. We also define the max and min distance as $\max_{i=1}^n D(\mathcal{P}_i, \mathcal{P}'_i)$ and $\min_{i=1}^n D(\mathcal{P}_i, \mathcal{P}'_i)$ respectively, denoted as $maxD(\mathcal{T}, \mathcal{T}')$ and $minD(\mathcal{T}, \mathcal{T}')$ \square

Our goal is to come up with a one-to-one mapping between each local trajectory and remote trajectory, such that the cost of this mapping is minimum with respect to the distances between those trajectory pairs. The semantic behind a mapping is that we “guess” the local trajectory and the remote one corresponding to the same individual. We use $AvgD$ instead of $minD$ or $maxD$ here, however, our algorithm can also adopt those two easily.

Since we do not have an identifier associated with a remote trajectory, we need to assign each local trajectory to a remote one based on certain criteria. Such an assignment needs to satisfy the

following requirement: each local trajectory is assigned to *at most* one remote one and each remote trajectory has *at most* one local trajectory assigned to it.

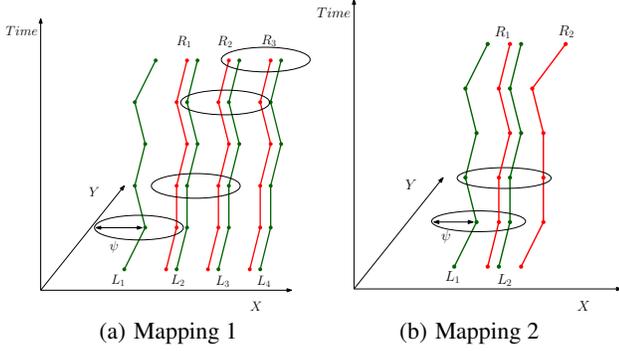


Figure 7: Example of Mapping Trajectories

In addition to the above requirement, we also need to make sure a local trajectory is assigned to a remote one which is closest to it. However, it is not trivial. Consider the following two scenarios in Figure 7(a) and Figure 7(b): in Figure 7(a) we have four local trajectories (L_1, L_2, L_3, L_4) marked in green, and three remote ones (R_1, R_2, R_3) marked in red. If we start from L_1 to look for a candidate mapping, there is only one possible assignment for L_1 which is R_1 . Since R_1 is already taken, L_2 will be assigned to R_2 and so forth. In the end, we get a mapping as follows: $\{(L_1, R_1), (L_2, R_2), (L_3, R_3)\}$. However, this is not the optimal solution. In the figure, we can see that the distances between L_{i+1} and R_i where $i = 1, 2, 3$ are very small. So a better mapping could be $\{(L_2, R_1), (L_3, R_2), (L_4, R_3)\}$. A greedy algorithm which pairs a local to a remote trajectory with least average distance is able to find such mappings for the example in Figure 7(a). However, will this solve our problem? Let us consider the following scenario shown in Figure 7(b).

From Figure 7(b), if we use a greedy algorithm to find the mapping, we will end up with the pair $\{(L_2, R_1)\}$ whereas L_1 will be mapped to R_2 which is a trajectory far away. A much better solution could be $\{(L_1, R_1), (L_2, R_2)\}$. The decision of whether a mapping is good or not really depends on users' preferences. For example, the user may try to find a reasonable mapping for each trajectory. In such a case, we may need to discard some mappings even though the local trajectory is very close to the remote one (e.g., the case in Figure 7(b)). However, if users want to have the closest mapping for one specific trajectory, some other local trajectories may be mapped to remote trajectories that are far away.

To solve the above problems, we propose a cost function between each pair of trajectories. Given a local trajectory \mathcal{T} and a remote trajectory \mathcal{T}' , the cost of mapping \mathcal{T} to \mathcal{T}' is as follows:

$$Cost(\mathcal{T}, \mathcal{T}') = \begin{cases} AvgD(\mathcal{T}, \mathcal{T}') & \text{if } AvgD(\mathcal{T}, \mathcal{T}') \leq \psi \\ W & \text{if } AvgD(\mathcal{T}, \mathcal{T}') > \psi \\ W & \text{if } \mathcal{T}' \text{ does not exist.} \end{cases}$$

where W is a system parameter assigned by the user which is in general greater than ψ . The above equation captures the semantic of each assignment. If the average distance between \mathcal{T} and \mathcal{T}' is less than ψ , then the cost is the average distance itself. Otherwise, the cost is equal to a user assigned parameter.

Given a local dataset with n trajectories and a remote dataset with m trajectories, if $n < m$ we introduce $m - n$ dummy local

trajectories, if $m < n$ we then introduce $n - m$ dummy remote trajectories. Since we introduced dummy trajectories, we need to modify the cost function as follows:

$$Cost(\mathcal{T}, \mathcal{T}') = \begin{cases} AvgD(\mathcal{T}, \mathcal{T}') & \text{if } AvgD(\mathcal{T}, \mathcal{T}') \leq \psi \\ W & \text{if } AvgD(\mathcal{T}, \mathcal{T}') > \psi \\ W & \text{if } \mathcal{T}' \text{ is dummy} \\ 0 & \text{if } \mathcal{T} \text{ is dummy.} \end{cases}$$

As an example, Figure 7(a) can be converted into the following cost matrix with W set to 10. Since the number of local trajectories is greater than number of remote trajectories, we introduce a dummy trajectory R_4 .

Table 1: Cost Matrix

	R_1	R_2	R_3	R_4
L_1	0.5	10	10	10
L_2	0.1	0.3	10	10
L_3	10	0.1	0.3	10
L_4	10	10	0.1	10

Given a local dataset D with n trajectories, a configuration C is defined as a one-to-one mapping between each local and remote trajectory. Our goal now is to minimize the following equation

$$Cost(C) = W * k + \sum_{i=1}^{n-k} Cost(\mathcal{T}_i, \mathcal{T}'_i) \quad (1)$$

where \mathcal{T}_i is mapped to \mathcal{T}'_i and k is the number of local trajectories mapped to dummy remote ones. This is an *assignment problem* [7], a well-known combinatorial optimization problem. It consists of finding a minimum weight matching in a weighted *bipartite graph*. By applying the cost function and introducing dummy trajectories in either local dataset or remote dataset, we can reduce our problem to an assignment problem.

Our optimization problem can be expressed as an integer program with the objective function:

$$\sum_{i \in L} \sum_{j \in R} Cost(i, j) * x_{ij}$$

subject to the constraints:

$$\begin{aligned} \sum_{j \in R} x_{ij} &= 1 \quad \text{for } i \in L \\ \sum_{i \in L} x_{ij} &= 1 \quad \text{for } j \in R \\ x_{ij} &\in \{0, 1\} \quad \text{for } i \in L, j \in R \end{aligned}$$

where $Cost(i, j)$ is the element in the i -th row and j -th column of the cost matrix. The above optimization problem is essentially a 0-1 integer programming or binary integer programming (BIP) problem [16]. Since our constraint matrix is totally unimodular and the right-hand sides of the constraints are integers, there exists a polynomial algorithm with $O(n^3)$ complexity, the details of which can be found in [20].

3.2.1 Adjusting Trust Scores

After the assignment procedure, the next step is to adjust trust scores based on each mapping. Given a local trajectory \mathcal{T}_L and a remote trajectory \mathcal{T}_R mapped to each other, we use the following cases to adjust the trust scores of each position $\mathcal{P} \in L$.

- if $\mathcal{P} \in \mathcal{T}_L$ and $\mathcal{P}' \in \mathcal{T}_R$ and \mathcal{P} is supported by \mathcal{P}' , we adjust the conflict and support analysis score $S_{cas}(\mathcal{P})$ of \mathcal{P} as follows:

$$S_{cas}(\mathcal{P}) = 1 - (1 - S_{init}(\mathcal{P})) \cdot (1 - S_{init}(\mathcal{P}')) \cdot \left(1 - \frac{D(\mathcal{P}, \mathcal{P}')}{\psi}\right)$$

- if \mathcal{P} conflicts with \mathcal{P}' , we adjust the conflict and support analysis score $S_{cas}(\mathcal{P})$ of \mathcal{P} as follows:

$$S_{cas}(\mathcal{P}) = S_{init}(\mathcal{P}) \cdot (1 - S_{init}(\mathcal{P}')) \cdot \frac{D(\mathcal{P}, \mathcal{P}') - \psi}{D(\mathcal{P}, \mathcal{P}') - \psi + 1}$$

Assuming that \mathcal{P} and \mathcal{P}' are independent events, the intuition behind these two functions is that if two positions \mathcal{P} and \mathcal{P}' support each other, the probability of \mathcal{P} being true is one minus the probability of them all being false. On the other hand, if two conflict each other, the probability of \mathcal{P} being true is the probability of \mathcal{P} being true times the probability of \mathcal{P}' being false. Note that we should also consider the distance between \mathcal{P} and \mathcal{P}' . We take it into account by introducing two factors, $1 - \frac{D(\mathcal{P}, \mathcal{P}')}{\psi}$ and $\frac{D(\mathcal{P}, \mathcal{P}') - \psi}{D(\mathcal{P}, \mathcal{P}') - \psi + 1}$. Recall that ψ is used to decide whether two positions support each other or not, these two factors range from 0 to 1. When $D(\mathcal{P}, \mathcal{P}') = \psi$, both factors evaluate to 0, which makes the trust score of \mathcal{P} unchanged.

3.3 Combining Trust Scores

We have seen how to calculate trust scores resulting from Common Pattern Analysis and Conflict/Support Analysis. We indicate those two trust scores for a position \mathcal{P} by $S_{common}(\mathcal{P})$ and $S_{cas}(\mathcal{P})$ respectively. We indicate the final trust score by $S(\mathcal{P})$. We obtain $S(\mathcal{P})$ by simply computing a weighted average of two scores as follows:

$$\begin{aligned} S(\mathcal{P}) &= w_{common} \cdot S_{common}(\mathcal{P}) \\ &+ w_{cas} \cdot S_{cas}(\mathcal{P}), \\ &\text{where } w_{common} + w_{cas} = 1.0. \end{aligned}$$

The weighting factors w_{common} and w_{cas} can be determined by many factors (e.g., quality of anonymized dataset, types or characteristics of trajectories, and so forth).

4. EXPERIMENTS

We report the empirical evaluation we have conducted in order to show the effectiveness and efficiency of the proposed algorithms. We developed a Java prototype, and all experiments were performed on a Intel(R) Core2 2.66GHz workstation with 4GB memory, running Windows Vista. The dataset used in our experiments is a pre-processed Truck dataset¹, which had also been used in [13, 4]. The difference between two consecutive time stamps in a trajectory is set to 300 seconds after pre-processing. We also cut trajectories when the difference in time between two consecutive observations is more than 600 seconds. In addition, we assume each trajectory starts at the beginning of a day so the first time stamp of each trajectory is 0. The reason why we re-align the trajectories is that by doing this the size of each equivalence class produced would be larger. Thus we can apply NWA algorithm to the dataset with larger k 's. We refer to the pre-processed dataset as *real dataset D*. There are 17,320 points and 2,675 trajectories in *D*. Another thing to note is that δ and ψ are measured in meters and 1 unit of ϵ represents 300 seconds on the plane of start time and end time.

¹Download at <http://www.rtreeportal.org>

4.1 Common Pattern Analysis

We use two datasets for this experiment: 1) *dataset_l* is a subset taken from the real dataset *D*; 2) *dataset_r* is the anonymization result by applying algorithms in [4] to the real dataset *D*.

Parameter	Setting
Completeness Ratio	20% 40% 60% 80% 100%
k	5 10 20 30 40 50
δ	400 600 800 1000 2000
ρ	3494 (5% of the diagonal)
σ	1.2
ϵ	5
α	0.0001
γ	0.001

Table 2: Parameters for Common Pattern Analysis

Table 2 lists the parameters used in the experiment. *Completeness Ratio* indicates the percentage of trajectories in real dataset that are put into *dataset_l*. In other words, if the completeness ratio is 10%, the probability that a trajectory in the real dataset will be put into *dataset_l* is 0.1.

Figure 8 (a) shows how completeness ratio affects the average score of local movements in *dataset_l* before and after taking in the anonymized data from *dataset_r*, when $k = 5$ and $\delta = 2000$. In the figure, the trust score reports the average score of the movements in *dataset_l*. We can easily tell that by using the anonymized *dataset_r*, the trust scores of local movements have been improved. Another thing to note is that the average score decreases when completeness ratio increases. One of the reasons is that there will be more movements (as well as patterns) included when the data is more and more complete. If many of them are related to each other, e.g., a lot of them share the same source cluster or the same destination cluster, then the average score would decrease. This reflects the rationale behind our score function for a pattern. Figure 8 (b) also shows how the completeness ratio affects the running time of our common pattern analysis algorithm.

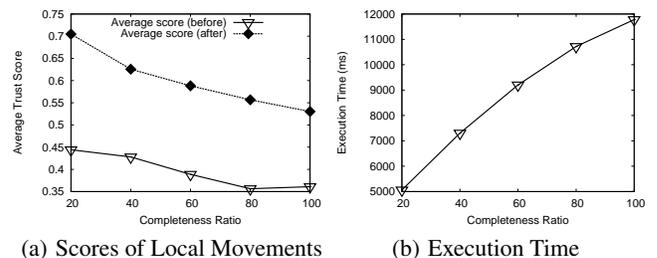


Figure 8: Varying Completeness Ratio

Figure 9 (a) and 9 (b) show the effects of changing k and δ on the average trust score for local movements. From Figure 9 (a), we can see the increase of score is less obvious when k is larger. It is intuitive since a larger k would imply a larger translation error in the procedure of anonymization and thus the quality of the anonymized data is lower. Figure 9 (b) indicates that when δ increases, the increase of average trust score is more obvious. This is also intuitive since a larger δ would imply a lower translation error in the procedure of anonymization and thus the quality of the anonymized data is higher.

4.2 Conflict/Support Analysis

We use two datasets for this set of experiments: 1) *dataset_l* is taken from the real dataset with distortion; 2) *dataset_r* is a subset

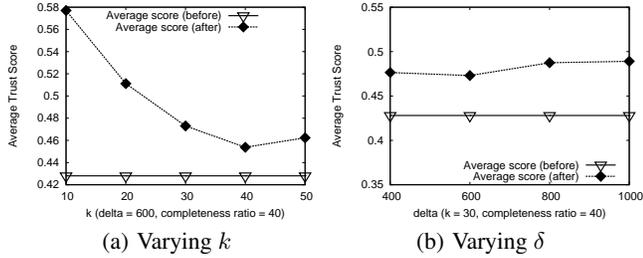


Figure 9: Varying Anonymization Parameters

of the anonymization result by applying algorithms in [4] on the real dataset. Different from the previous setting, we choose a certain number of trajectories from the real dataset as our $dataset_l$. Doing so reflects reality, since the number of suspects is rather small in practice. As to $dataset_r$, it is the union of all equivalent classes which share the same time span with trajectories in $dataset_l$. The initial trust scores of both $dataset_l$ and $dataset_r$ are 0.5.

Parameter	Setting
Number of Suspects	10 20 30 40 50
Distortion Ratio	10% 20% 30% 40% 50% 60% 70% 80% 90%
k	5 10 20 30 40
δ	200 400 600 1000 2000
Distortion Percentage	5% 10% 15% 20% 25% 30%
ψ	5% 10% 20%
Initial Trust Score	0.5

Table 3: Parameters for Conflict and Support Analysis

Table 3 lists the parameters used in this experiment, where values in bold denote default values. *Number of Suspects* indicates trajectories that are put into $dataset_l$. *Distortion Ratio* controls how many positions in $dataset_l$ are maliciously modified. *Distortion Percentage* controls how much error we introduce to a single position. The percentile values represents the percentage of the dataspace (the diagonal of the rectangle that contains all trajectories). We refer to local data that have been modified as *incorrect* and the others as *correct*.

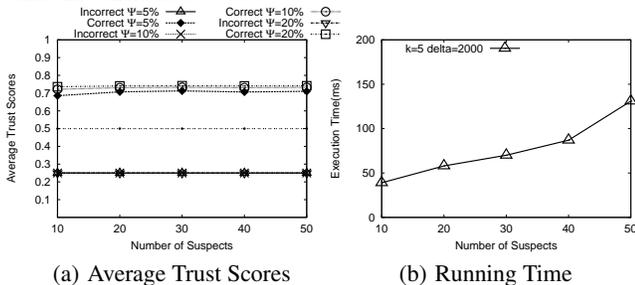


Figure 10: Varying Number of Suspects

Figure 10 (a) reports the average trust scores of both *correct* and *incorrect* data for different number of trajectories in $dataset_l$. We can observe clearly that our approach is able to distinguish those incorrect data from correct ones. The trust scores of correct data are consistently higher than the incorrect ones, and very close to 0.75, which is the highest score achievable in this scenario. The trust scores of incorrect data are close to 0.25 which, on the other hand, is the lowest possible value. We can also observe that the average scores of correct data increase slightly when ψ increases. This is because if two positions are considered to support each other, the greater ψ , the higher the trust score. Figure 10 (b) reports the

execution time with the change of number of local trajectories. Although the execution time increases when the number of trajectories increases, the total running time is still very low.

Figure 11 (a) reports the average trust scores when changing the distortion percentage. Note that: 1) when distortion percentage is small, the trust score of incorrect data is close to the correct data, however, we are still able to distinguish between them especially for small ψ value (e.g., $\psi = 5\%$); 2) When distortion percentage grows, the gap between correct data and incorrect data is widening. This is because when distortion is larger, it is easier to identify incorrect data; 3) There is a remarkable drop of trust score for each different ψ value, this is because when the distortion percentage is bigger than ψ , suddenly those data will be considered as conflicting with their remote counterparts; 4) The trust scores of correct data remain almost constantly high, which demonstrates the robustness of our approach.

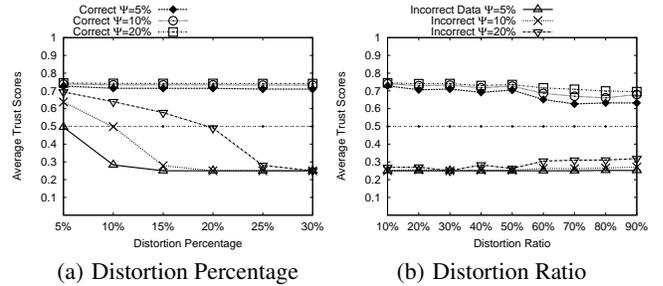


Figure 11: Varying Distortion Percentage and Ratio

Figure 11 (b) reports the average trust scores by varying the number of modified data. We can observe that our method is robust: even when 90% of data are being modified, our method can still distinguish correct data and incorrect data very easily.

Figure 12 (a) reports how the change of δ values affects the average trust scores ($k = 40$). The scores of incorrect data are almost constant and close to 0.25. The average scores of correct data increase slightly when δ increases, because the space distortion becomes smaller when δ is large.

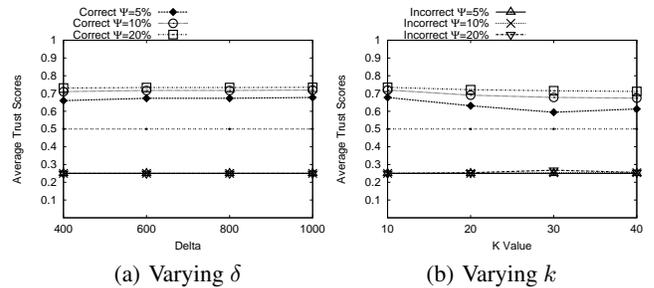


Figure 12: Varying Anonymization Parameters

Figure 12 (b) reports how the change of k values affects the trust scores ($\delta = 1000$). Incorrect scores still remain constant, however, the correct scores decrease when k increases. This is because a large k introduces more space distortion. Figures 12 (a) and 12 (b) demonstrate that large k and small δ introduce more space distortion and thus reduce the quality of the data.

5. RELATED WORK

Work related to our approach falls into three categories: trustworthiness computation, uncertainty in moving objects, and location privacy.

In [10], an approach is proposed for computing the trustworthiness of location data based on provenance. They use intra/inter-source assessment along with the provenance information to assign trust scores for location data. However, their approach cannot be applied to anonymized data. A conceptual framework is proposed in [6] for computing the trust scores of both data items and data sources. Trust scores are affected by four factors: data similarity, data conflict, path similarity, and data deduction. Based on the framework, an approach for computing trust scores of data items and data sources is also proposed in [11]. However, it does not deal with location data. The work in [13] proposed pattern mining for trajectories, which is similar to the approach we used in our common pattern analysis. However, their approach cannot be applied to anonymized data. To the best of our knowledge, our method is the first that studies privacy-preserving trustworthiness computation.

Uncertainty problems in moving object databases have been well-investigated [22]. Uncertainty is an inherent aspect in trajectories of moving objects since the location of a moving object is not always precise due to the continuous motion and network delays [21]. Pfooser and Jensen [18] presented a formal quantitative method to handle uncertainty in moving objects. Recently, [9] gave a probabilistic approach for estimating the answer to a few categories of queries over uncertain values of dynamic data. These approaches, however, can only handle precision issues, but not trustworthiness issues. In contrast, our work addresses trustworthiness of trajectory data.

The topic of *location k-anonymity* for LBS has been extensively studied [14, 12, 17, 5, 15]. The basic idea is that a message sent from a user is *k*-anonymous when it is indistinguishable from the spatial and temporal information of at least *k* - 1 messages sent from other users. In [4], the authors studied how to publish trajectories such that anonymity of the individuals is preserved, while at the same time maintaining good data quality. We adopted their privacy model and developed our approach based on it.

6. CONCLUSION

In this paper, we illustrated the importance of taking into account privacy when computing trustworthiness of location data in applications such as forensics. We proposed two privacy-preserving techniques for *common pattern analysis* and *conflict/support analysis* that can improve trustworthiness of location data with the help of large repositories of anonymized trajectories. An extensive experimental evaluation demonstrated the effectiveness and efficiency of our approach.

In future work, we plan to investigate how to extend our techniques to other privacy-preserving paradigms, such as differential privacy, a model that has gained a lot of traction in recent years. Also, we plan to research mechanisms for privacy-preserving processing of provenance, which can further provide insights into the level of data trustworthiness based on the reliability of data sources.

7. ACKNOWLEDGMENTS

The work reported in this paper has been partially supported by NSF under grants CNS-1111512 and CNS-0964294.

8. REFERENCES

- [1] <http://www.cdc.gov>. *Center For Disease Control and Prevention*, 2009.
- [2] consumercal.live.radicaldesigns.org/article.php?id=1395. *SB 1268 Support Letter*, 2011.
- [3] <http://www.bitcarrier.com/>. *Bitcarrier*, 2011.
- [4] O. Abul, F. Bonchi, and M. Nanni. Never walk alone: Uncertainty for anonymity in moving objects databases. In *ICDE*, pages 376–385, 2008.
- [5] A. R. Beresford and F. Stajano. Mix zones: User privacy in location-aware services. In *PerCom Workshops*, pages 127–131, 2004.
- [6] E. Bertino, C. Dai, H.-S. Lim, and D. Lin. High-assurance integrity techniques for databases. In *BNCOD*, pages 244–256, 2008.
- [7] R. Burkard, M. Dell’Amico, and S. Martello. *Assignment Problems*. SIAM, 2009.
- [8] S. S. Chan, D. Leung, H. Chui, A. F. Tiwari, et al. Parental response to child’s isolation during the sars outbreak. *Ambulatory Pediatrics*, 7(5), 2007.
- [9] R. Cheng, D. Kalashnikov, and S. Prabhakar. Evaluating probabilistic queries over imprecise data. In *SIGMOD*, pages 551–562, 2003.
- [10] C. Dai, H.-S. Lim, E. Bertino, and Y.-S. Moon. Assessing the trustworthiness of location data based on provenance. In *GIS*, pages 276–285, 2009.
- [11] C. Dai, D. Lin, E. Bertino, and M. Kantarcioglu. An approach to evaluate data trustworthiness based on data provenance. In *Secure Data Management*, 2008.
- [12] B. Gedik and L. Liu. Location privacy in mobile systems: A personalized anonymization model. In *ICDCS*, pages 620–629, 2005.
- [13] F. Giannotti, M. Nanni, F. Pinelli, and D. Pedreschi. Trajectory pattern mining. In *KDD*, pages 330–339, 2007.
- [14] M. Gruteser and D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *MobiSys*, 2003.
- [15] M. Gruteser and X. Liu. Protecting privacy in continuous location-tracking applications. *IEEE Security & Privacy*, 2(2):28–34, 2004.
- [16] J. K. Karlof. *Integer programming: theory and practice*. CRC Press, 2006.
- [17] H. Kido, Y. Yanagisawa, and T. Satoh. Protection of location privacy using dummies for location-based services. In *ICDE Workshops*, page 1248, 2005.
- [18] D. Pfooser and C. Jensen. Capturing the uncertainty of moving objects representation. In *Int’l Symp. on Advances in Spatial Databases*, pages 111–132, 1999.
- [19] K. Pye and D. Croft. *Forensic geoscience: Principles, techniques and applications*. Springer Berlin / Heidelberg, 2005.
- [20] A. Schrijver. *Combinatorial optimization: Polyhedra and Efficiency*. Springer, 2003.
- [21] G. Trajcevski, O. Wolfson, K. Hinrichs, and S. Chamberlain. Managing uncertainty in moving objects databases. *ACM Trans. Database Syst.*, 29(3):463–507, 2004.
- [22] O. Wolfson, S. Chamberlain, S. Dao, L. Jiang, and G. Mendez. Cost and imprecision in modeling the position of moving objects. In *ICDE*, pages 588–596, 1998.