Identifying Unsafe Routes for Network-Based Trajectory Privacy

Aris Gkoulalas-Divanis[*] Vassilios S. Verykios[†] Mohamed F. Mokbel[‡]

Abstract

In this paper, we propose a privacy model that offers trajectory privacy to the requesters of Location-Based Services (LBSs), by utilizing an underlying network of user movement. The privacy model has been implemented as a framework that (i) reconstructs the user movement from a series of independent location updates, (ii) identifies routes where user privacy is at risk, and (iii) anonymizes online user requests for LBSs to protect the requester for as long as the service withstands completion. In order to achieve (iii), we propose two anonymization techniques, the K*–present* (weak) and the K*–frequent* (strong) trajectory anonymity, and a second chance approach that takes over when anonymization fails to ensure that the privacy of the user is preserved. To the best of our knowledge, this is the first work to propose a trajectory privacy model that utilizes an underlying network of user movement to offer in an interactive way personalized privacy to online user requests on trajectory data.

1 Introduction

The technological advances in sensors and wireless communications, along with the advances in the telecommunication industry have made possible the offering of high accuracy in location tracking at an affordable cost. The increased location accuracy gave rise to a series of location-based applications, the so-called Location-Based Services (LBSs), that exploit positional data to offer high-end services to their subscribers. The new computing paradigm is changing the way people live and work but also poses a series of challenges as it touches upon delicate privacy issues. Without strict safeguards, the deployment of LBSs and the sharing of location information may easily lead the way to an abuse scenario. As an effect, sophisticated algorithms that offer privacy to the users of LBSs have to be devised.

A wide variety of algorithms has been recently proposed for the offering of privacy in LBSs (e.g., [3, 6, 10, 11, 12, 16]). Although some of the proposed approaches assume a decentralized scenario where the mobile devices of the users cooperate to offer privacy to the requester (e.g., [7, 8]), the majority of existing work is based on the centralized scenario of a trusted server. This scenario has certain benefits and is also employed in the current work. It can be described as follows: We consider a population of mobile users who are supported by some telecommunication infrastructure, owned by a telecom operator. Every user owns a mobile device that periodically transmits a location update to some traffic monitoring system residing in a trusted server of the telecom operator. A set of LBSs are available to the subscribed users through service providers that collaborate with the telecom operator. We assume that these service providers are not trusted; if a user submits a request for an LBS directly to the service provider then her identity can be revealed and her privacy can be compromised. Motivated by this fact, the centralized scenario requires that every user request for an LBS has to be submitted to a trusted server of the telecom operator. The role of the trusted server is to filter the incoming user requests and to produce anonymous equivalents that can be safely forwarded to the (unsafe) service providers in order to be serviced. To produce the anonymous equivalent to an original user request, the trusted server has to incorporate algorithms that (i) remove any obvious identifiers that are part of the user request (e.g., user ID, name, etc) and (ii) effectively transform the exact location of request into a spatiotemporal area that includes a sufficient number of other users in the system to prevent the attacker from locating the requester.

The transformation of the exact user location (x, y) at the time of request t to a spatiotemporal area $(A(x, y), [t_1, t_2])$ is achieved through the use of (spatial) K–anonymity. The K–anonymity principle for relational data [17, 18] requires that each record in a given dataset is indistinguishable from at least $K - 1$ other records with respect to a certain set of identifying variables, known as the quasi-identifier. In the context of LBSs, the K–anonymity principle requires that the spatiotemporal area that is produced by the trusted server from the exact location of the user request is such that the identity of the requester cannot be disclosed with a probability that is larger than $1/K$, among $K - 1$ other users. Existing work on privacy in LBSs fails to address one or more real-world challenges as (i) it does not consider that users typically move in a network-confined environment (e.g., a road network), which allows for certain decisions to be made with respect to the offering of privacy, (ii) all user requests are anonymized in the same manner no matter what the location of the user is at the time of request,

[*]Department of Computer and Communication Engineering, University of Thessaly, Volos, Greece. Email: arisgd@inf.uth.gr

[†]Department of Computer and Communication Engineering, University of Thessaly, Volos, Greece. Email: verykios@inf.uth.gr

[‡]Department of Computer Science and Engineering, University of Minnesota, Minneapolis, USA. Email: mokbel@cs.umn.edu

as well as her future locations until the provision of the service, and (iii) it does not build on existing DBMSs that have numerous advantages over ad-hoc/"from scratch" implementations: data independence and efficient access; reduced time for maintenance; querying support for a vast amount of data; data integrity and security; uniform data administration; concurrent access, recovery from crashes, and fault tolerance.

In this paper we improve current state-of-the-art algorithms for the offering of privacy in LBSs by providing a solution that addresses all the above mentioned challenges. Specifically, to address the first challenge, we make use of the network modeling capabilities offered by modern DBMSs, which allow us to capture user movement within an underlying network. To address the second challenge, we utilize the history of location updates for every user in the system (as collected by the trusted server) to build representative patterns of user movement. We argue that a traffic pattern can be used to breach user privacy when it reaches an outside party, even if it lacks of any user identification data. To motivate this argument, think of a scenario where a user goes by certain city areas when she commutes to work in more or less the same times in weekdays. This frequent behavior of the user can lead to a possible identification simply by matching either the origin of her trip to some public domain geocoded information for the house, or the destination to the location of the business facility that this user works. Furthermore, a possible matching of a series of location updates with a frequently traveled route of a user may also easily lead to her identification. To adequately protect the privacy of the user, we utilize the identified traffic patterns to deliver to the user the most appropriate between two types of trajectory K–anonymity. Finally, to address the third challenge, we implement our proposed solution on top of Oracle, while we discuss its possible implementation on top of other modern DBMSs.

In our privacy model we assume an attacker who has knowledge of the following: (i) the frequent movement behavior of all the users in the system, computed by the trusted server as part of its functionality, (ii) the location updates of the users, (iii) the anonymized versions of all the requests transmitted by the trusted server to the service providers, and (iv) the algorithms used by the trusted server to support user privacy. The proposed privacy model is implemented as a framework that comprises of four phases. In the first phase, the trusted server uses the collected location updates, along with the underlying network of user movement, to reconstruct the (recent) history of movement for each user in the system. In the second phase, the reconstructed user movement is used to identify the frequent movement behavior of each user. Frequent movement behavior is captured as either safe or unsafe with respect to user privacy, based on whether it is also ordinary or atypical for a significantly large number of nearby users. In the third phase, we provide the

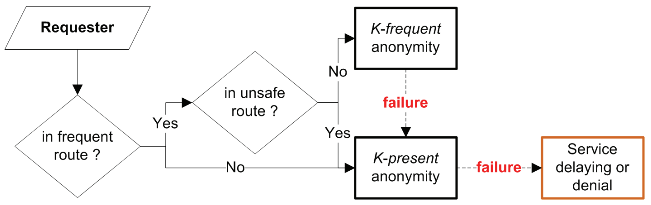


Figure 1: Anonymity strategies in the network aware model.

user with trajectory K–anonymity for handling online user requests on trajectory data. K–anonymity is essential to protect the privacy of the users, starting from the point of request for a service and continuing for as long as the requested service withstands completion. As part of our framework, we deliver two types of trajectory K–anonymity. *K–present* (weak) trajectory anonymity identifies K–1 subjects that are close to the requester at the time of request and thus could have issued the request (from the viewpoint of the service provider). On the other hand, *K–frequent* (strong) trajectory anonymity collects the subjects that are near the requester at the time of request and for whom the current route of the requester is also frequent. As shown in Fig. 1, the choice of the anonymity strategy depends on the location of the requester at the time of request and her subsequent locations until the completion of the service provision. In the offering of trajectory K–anonymity, our model takes special care to protect the requester from sequential tracking. In sequential tracking, the attacker examines the anonymity regions, produced for the protection of the user as part of an on-going service, to locate subjects that participate in only some of these regions. Since the requester has to be part of all the anonymity regions computed for her protection, through the process of elimination, the confidence of the attacker regarding the identity of the requester can substantially increase. The final phase of the proposed framework deals with the event of failure in the provision of trajectory K–anonymity. In this case, the trusted server postpones the servicing of the user request for a small period of time. After that, if the anonymization process fails again, the requester is protected by blocking the servicing of the request.

The proposed privacy framework relies on a user privacy profile that stores the necessary information related to the privacy requirements of the user. This includes (i) the preferred value of K (in K–anonymity) for each requested LBS, (ii) (a pointer to) the frequent routes of the user, and (iii) the minimum spatial area A_{min}, around the requester, where the participants of the anonymity set should be searched for so that the user is adequately covered up. This threshold defines the minimum extend of the spatial area that must replace the real location of the user, in the anonymized request. Apart from A_{min} the system has knowledge of the coarsest spatial resolution A_{max} required for the provision of each LBS.

This is a system-defined parameter that provides the maximum area of the anonymity set (with respect to the requested service) in order to ensure that the service can be delivered.

The contributions of this work are as follows:

- We propose a privacy model that utilizes an underlying network that confines user movement to deliver, in an interactive way, different levels of user privacy to online user requests in trajectory data.
- By using the movement history of the users in the system, we automatically extract patterns depicting the frequent movement behavior of each user. By contrasting these patterns with those of the rest of the population in the system, we classify each of them as either safe or unsafe with respect to the privacy of the user.
- We propose a privacy model that utilizes the unsafe patterns of the requester to deliver the most appropriate between two alternatives for trajectory K–anonymity, while protecting the user from sequential tracking.

The remainder of this paper is structured as follows. Section 2 presents the related work, while Section 3 introduces the necessary terminology. In Section 4, we present the algorithms that support the proposed privacy methodology. Section 5 presents a set of issues that pertain to the implementation of the system framework. Section 6 contains the experimental evaluation and Section 7 concludes this work.

2 Related work

Several approaches have been recently proposed for the sanitization of trajectory data [1, 2, 13, 19]. All these approaches aim at constructing a privacy-aware version of some collected trajectory data, to allow for data publication. On the other hand, our model offers privacy in online requests for trajectory data. For this reason, our methodology lies closer to algorithms offering location privacy in LBSs than to offline approaches for trajectory privacy. Thus, in what follows, we review state-of-the-art work in the area of location privacy in LBSs, contrasting it to our methodology.

Gruteser and Grunwald [10] anonymize location data through the use of a spatial and a temporal cloaking strategy. For the spatial cloaking, the total area covered by the anonymizer is recursively subdivided into equi-size quadrants, until the quadrant where the user is located contains at least K–1 other subjects. The temporal cloaking incurs after the spatial cloaking and delays the servicing of the request until the anonymity requirements are met.

Bettini, et al. [3] keep track of the personal history of location updates of each user of LBSs, along with a sequence of spatiotemporal patterns that act as a pseudo-identifier for this particular user. Each pattern involves an area and a time span, while the sequence of the patterns is accompanied by a recurrence formula defining the minimum number of observations that led to its characterization as hazardous for the privacy of the user. Although this work sets a new perspective in the provision of privacy to location data, it lacks the reporting of any results.

Mokbel, et al. [16] present a privacy-aware query processing system that offers location K–anonymity. A partitioning approach similar to [10] is applied, with the entire area of the anonymizer being divided in a grid fashion and organized in a pyramid structure. Then, a bottom-up anonymization algorithm iterates over the different layers of the pyramid to identify the aggregate cells that capture the requester along with her neighbors and satisfy both K–anonymity and the minimum covered area.

Gruteser and Liu [11] attack the anonymity problem in the context of disclosure-control by proposing a user-driven partitioning of the area covered by the LBSs into safe and sensitive regions. Whenever the user is located within one of her sensitive regions, the system defers from providing location updates to the service providers. Furthermore, the proposed anonymity strategy ensures that no location updates are released that would give away which, of at least K sensitive areas, the user visited.

Kalnis, et al. [12] present the nearest neighbor cloaking algorithm that generates anonymity sets where the requester is expected to be located far from the center of the anonymity region. To achieve that, it determines the K–1 nearest neighbors of the requester and then randomly selects one of them to formulate a new region that contains its K–1 nearest neighbors. The region of anonymity is then computed as the MBR of the second region, expanded if necessary to include the requester.

An approach for the mining of frequent trajectory patterns ($\mathcal{T}$–patterns) was recently proposed by Giannotti, et al. [9]. The trajectories are modeled as temporally annotated sequences capturing a set of elements (the places of interest) along with typical transition times to move from one element to another. An example of a $\mathcal{T}$–pattern is Home $\xrightarrow{15min}$ Park $\xrightarrow{30min}$ Work. The authors provide algorithms both for the identification of the places of interest and the mining of frequent $\mathcal{T}$–patterns.

Our paper follows the widely adopted paradigm of an intermediate trusted server (e.g., [10, 16]) that handles all the sensitive location data, leaving the service providers to deal only with anonymous data. Instead of delaying the servicing of the requests ([10]), our proposed methodology relies on the recent history of user location updates to identify subjects that were near the requester within a reasonable amount of time and who could have initiated the request. Following [3], our model keeps track of the movement history of each user in the system. However, we capture the routes that were traveled by the users, instead of storing the sequence of received location updates. Our approach partitions the history of location updates into a set of safe and unsafe routes per user, but unlike [11] these regions have also a temporal dimension and correspond to actually traveled routes. This is also the difference of our work to [9] since our model captures the particular route that the user has followed between any two consecutive "places of interest", discriminating among alternative routes to move from one location to another. As in [12] our proposed anonymization approach takes care in positioning the requester far from the center of the computed anonymity region, ensuring that he or she cannot be identified with a probability that is higher than $1/K$. However, unlike [12], it achieves that without paying the extra cost of searching for more neighbors than the minimum required. Finally, this is the first work to propose a methodology for the provision of personalized trajectory K–anonymity in a network-confined environment and to offer a network aware solution for the automatic extraction of the safe and the unsafe routes of the users. To our knowledge, this is also the first work to build a privacy framework for LBSs on top of a modern spatial DBMS.

3 Terminology

Let o denote a moving object. A *location update* is a tuple (o, x, y, t) stating that user o was located at (x, y) at time t (Fig. 2(a)). The trusted server collects for each subscribed user, his or her sequence of location updates and uses this information to reconstruct the user *history of movement*. However, this information cannot be directly used to draw any significant conclusions regarding the movement habits of the user. For this reason, the movement history of each user needs to be decomposed into smaller blocks, the trajectories. The decomposition of the movement history into trajectories is necessitated by the need to identify frequent movement patterns of users, having the most interesting granularity for the application at hand. A *trajectory* of a moving object o is a part of the movement history of this object that consists of a sequence of consecutive location updates, beginning at some instance $(x_{start}, y_{start}, t_{start})$ and terminating at another instance $(x_{end}, y_{end}, t_{end})$ (Fig. 2(c)). The decomposition of the movement history to a set of trajectories, takes place by identifying stops that denote the immobility of the object for a sufficiently large period of time, indicated by the system parameter t_{stop} (e.g., half hour). Having identified such stops, the reconstructed history of user movement (captured as a compound 3D polyline that approximates real user movement) that is bounded by two consecutive stops, corresponds to a trajectory. The time period of a stop is properly adjusted to avoid accounting for delays that are too short (i.e., the stop of a car in a road intersection or in the traffic lights) and may cause unwanted segmentation. The decomposition of the user history of movement into trajectories is application-specific, meaning that different applications may

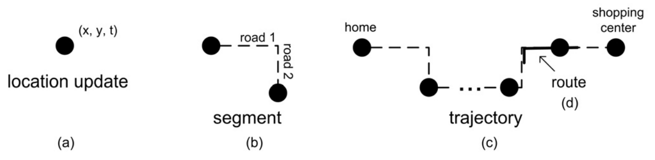


Figure 2: The building blocks of user movement.

require a different value of t_{stop} to handle different types of moving objects (e.g., pedestrians, cars) and generate meaningful trajectories. Finally, a trajectory consists of segments, where a *segment* is the reconstructed history of user movement that connects two consecutive location updates and defines the itinerary that the user followed (Fig. 2(b)). Any part of a trajectory, independently of its start point and end point, is called a *route* (Fig. 2(d)).

The trajectories allow us to expose some interesting knowledge for the user by tracking regularities in her movement. In the proposed privacy model, a trajectory is said to be *valid* when all its segments define actual itineraries in the underlying network (i.e., are consistent to the network) and has a concrete start and end point. To protect the privacy of the user when submitting requests for LBSs, the system has to identify routes of the user that are frequent. These are called *frequent routes* and are defined as follows:

DEFINITION 3.1. ***(Frequent route)*** *A route of a moving object o is defined as frequent if it appears among the trajectories of o a number of times that is larger than a minimum (system-defined) frequency threshold $freq$.*

The number of times that the route appears in the trajectories of o is called the *frequency* of the route. Furthermore, any route that is not frequent is called *infrequent*. Directly related to the notion of a frequent route are the definitions of a safe and an unsafe route.

DEFINITION 3.2. ***(Safe route)*** *A route of a moving object o is considered as safe if it is frequent for o and also frequent for at least K–1 other moving objects in the system.*

Similarly, a route of a moving object o is considered as *unsafe* if it is frequent for o and also frequent for at most K–2 other moving objects in the system. A request of a moving object o for an LBS, is defined as follows:

DEFINITION 3.3. ***(Request for an LBS)*** *A request R_c for an LBS is a tuple in the form R_c = <o, sid, x, y, t, data>, where o is the requester, sid is the identifier of the requested service, (x, y) is the location of object o at the time of request t, and data is any necessary service-specific information.*

Depending on the user location and time when requesting an LBS and the consecutive locations and times when she submits location updates until the service is provided, the proposed approach delivers two types of trajectory K–anonymity[1]. In what follows, we define the notion of an anonymity set, as well as the two types of offered K–anonymity.

DEFINITION 3.4. *(**Anonymity set**) Given a moving object o sending a request R_c for an LBS, we define the anonymity set of o, to be (at least) K–1 objects that are close to o at the given point in time (who could have initiated request R_c).*

DEFINITION 3.5. *(**K-present anonymity**) A moving object o, when submitting a request, is K–present anonymous if each participant of its anonymity set could be a potential issuer of the request from within an area that is near to the requester and within a time period that is close to the time of request.*

DEFINITION 3.6. *(**K-frequent anonymity**) A moving object o, when submitting a request, is K–frequent anonymous if each participant of its anonymity set shares the same frequent route with object o and passes by near the location of the requester at a time period that is close to the time of request.*

4 Network aware privacy model

This section introduces the network aware privacy model for the offering of trajectory K–anonymity. Our model consists of four phases. Phase I reconstructs the history of user movement and decomposes it into trajectories. Phase II detects the frequent routes for each user and separates them into either safe or unsafe with respect to his or her privacy requirements. Phase III is responsible for the provision of trajectory K–anonymity. Finally, Phase IV offers a service delaying or denying mechanism that is employed to safeguard the privacy of the users when K–anonymity fails. In what follows, we consider a spatial network (graph) $\mathcal{N}(V, E)$ that models the real network topology of user movement through a set of nodes V and edges E. A node $n \in V$ represents a road intersection and an edge $(n, n') \in E$ represents the existence of a road part that connects nodes n and n'.

4.1 Phase I: Reconstruction of user movement Under the network aware model, movement is restricted to a series of routes that are consistent to the underlying network. The existence of inaccuracies in the location updates that are collected by the trusted server, as well as the existence of irregularities in the movement behavior of the users, make the process of movement history reconstruction complicated. Specifically, given any type of physical network (such as a road network), an object is rather rare to transmit the exact same sequence of location updates when following the same route at two different occasions (e.g., at two distinct days of the month). An example of this situation is highlighted in Fig. 3, where the user is shown to have crossed the same road links but has sent different location updates in each of the two trajectories (see the red polylines).

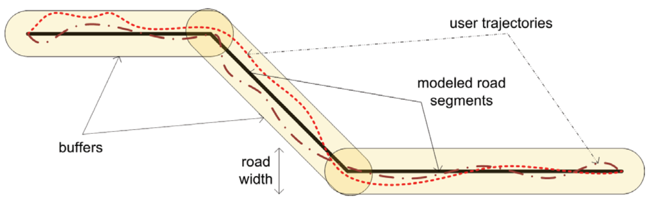


Figure 3: Possible motion curves and the role of buffers.

Algorithm 1 Reconstruction of user movement.

```
Input: Two consecutive location updates $A$, $B$.
Output: Reconstructed movement from location $A$ to location $B$.

function RECONSTRUCT($A(x_A, y_A, t_A)$, $B(x_B, y_B, t_B)$)
    INSERTTONET($A$); INSERTTONET($B$)                ▷ insert $A$, $B$ to the network
    Cand_routes $P \leftarrow$ GET_$m$_SHORTEST_ROUTES($A$, $B$, $m$)
    foreach $P_i \in P$ do
        $t_{AB,i} \leftarrow$ COMPUTE_TRAVEL_TIME($P_i$)
    end foreach
    REMOVEFROMNET($A$); REMOVEFROMNET($B$)            ▷ remove $A$, $B$
    return route having $argmin_i(|t_{AB,i} + t_A - t_B|)$
end function

procedure INSERTTONET($p(x_p, y_p, t_p)$)
    if $\nexists$ buffered geometry $G$ with $p \in G$ then
        $G \leftarrow$ GET_NEAREST_NEIGHBOR($p$)          ▷ nearest link
    end if
    $p \leftarrow$ PROJECT($p$, unbuffered $G$)
    Let $u$, $v$ be the end points of $G$
    set $G$ as inactive
    add edges $up$ and $pv$
    update mean_travel_time($up$, $pv$, $G$)   ▷ approx. time to traverse $up$ and $pv$
    set edges $up$ and $pv$ as active
end procedure

procedure REMOVEFROMNET($p(x_p, y_p, t_p)$)
    identify the edges $Q$, $R$ where $p \in Q$, $p \in R$
    get the other end points $u$, $v$ of the edges $Q$, $R$
    remove edges $Q$, $R$ and vertex $p$
    reset edge $uv$ to active
end procedure
```

Thus, a methodology is needed to deal with the uncertainty regarding the recording of an exact location of an object at a certain time. Research works, such as [14, 15] (among others), study the reconstruction of movement in a network-confined environment. Our approach is a simplified version of [14] that offers good computational speed and requires less data housekeeping. It is shown to perform well, by providing accurate reconstruction at a degree that is sufficient for the workings of the proposed anonymization technique. As part of our model, the links of the underlying net-

[1]Trajectory K–anonymity differs from the widely adopted location K–anonymity (see the approaches presented in Section 2) as it preserves the privacy of the requester for as long as the requested service withstands completion. This means that the whole user trajectory, starting from the time of request and for as long as the requested LBS is in progress, is protected from disclosure.

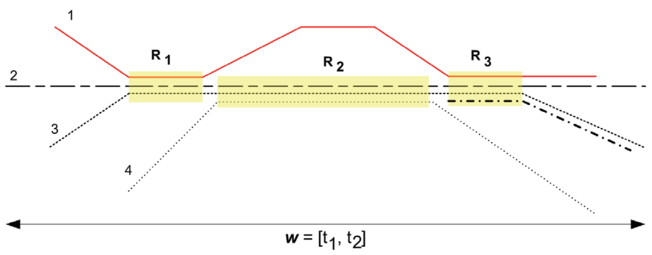


Figure 4: Frequent routes in the network aware model.

work (e.g., corresponding to a road or a road part) are considered to be cylindrical areas in the three-dimensional (3D) space, as shown in Fig. 3. This process is called *buffering* and has been first used in [20]. In our model, the radius of the cylinder defines the spatial extend of the link and depends on its topology in the underlying network. The advantage of this representation is that it abstracts away possible recording and measurement errors and identifies when two trajectories are the same, even if they do not coincide in space and time. Buffers allow us to map the collected points to the appropriate links of the network.

The reconstruction algorithm applies an incremental, edge–by–edge strategy that iteratively constructs the 3D polyline of the user history of movement from the corresponding location updates. Specifically, given two consecutive location updates the algorithm reconstructs the corresponding segment by identifying the most probable route that the moving object has followed through the (links of the) network. To achieve that, it compares the actual time that was needed by the moving object to cover the distance between two consecutive location updates, to the mean travel time as attained through the network (e.g., through estimates based on the current road traffic condition information). All links of the network have a buffer equal to the width of the modeled road (see Fig. 3). Algorithm 1 provides the details of our implementation. In the rare event that the collected user location update does not lie inside a buffered link of the network, we apply nearest-neighbors matching to capture the corresponding link. Some alternative strategies can be found in [4, 21].

After the reconstruction of the user movement history, the next step is its decomposition into trajectories. This is an application-specific process and thus knowledge is required regarding the appropriate value of t_{stop} with respect to the considered application. Assuming that we have this kind of knowledge, the decomposition of the user movement history into trajectories is a straight-forward process.

4.2 Phase II: Identification of frequent routes The knowledge of the trajectories of the users allows us to compute their frequent routes and to subsequently classify them as either safe or unsafe. The identification of the frequent

Algorithm 2 Derivation of the frequent routes.

Input: Object o along with all its trajectories tr; time interval $\boldsymbol{w}$ to search for frequent routes and frequency threshold *freq*.
Output: The frequent routes of o in time interval $\boldsymbol{w}$.

```
 1: procedure FREQUENTROUTES($o$, $tr$, $\mathcal{N}$, $\boldsymbol{w}$, $freq$)
 2:     $tr_w \leftarrow \{ tr \,|\, t \in \boldsymbol{w} \}$          ▷ parts of the trajectories that are in w
 3:     declare Hash $Count\{\} \leftarrow \varnothing$
 4:     foreach trajectory $T \in tr_w$ do
 5:         $L \leftarrow$ GET_OVERLAPS(geom($T$), link_geom($\mathcal{N}$))
 6:         foreach $l \in L$ do                                  ▷ for each link of $L$
 7:             $Count\{l.\text{linkid}\}$++                       ▷ keep track of the traversal
 8:         end foreach
 9:     end foreach
10:     declare List $F \leftarrow \varnothing$                   ▷ list of frequently visited links
11:     foreach $f \leftarrow key$ (Count) do
12:         if Count$\{f\} \geq freq$ then
13:             INSERT($F$, $f$)                                   ▷ $f$ is frequent
14:         end if
15:     end foreach
16:     $FR \leftarrow (F, w)$                                     ▷ frequent routes of $o$ in w
17: end procedure
```

routes of a user, proceeds as follows. Consider Fig. 4 where a set of trajectories is presented for a time interval $\boldsymbol{w} = [t_1, t_2]$. Assuming that all four trajectories belong to the same moving object o and the frequency threshold (see Definition 3.1) is 3, areas R_1, R_2 and R_3 correspond to frequently traveled routes for this object.

Algorithm 2 presents our implementation that derives the frequent routes of an object o when given *all* its trajectories in the system. The algorithm considers a pre-specified time interval $\boldsymbol{w}$ and retrieves all the trajectories (or routes) of user o that are time-consistent with respect to $\boldsymbol{w}$. A trajectory (or route) is time-consistent with respect to a given time interval $\boldsymbol{w}$, if all its spatiotemporal points (x, y, t) have $t \in \boldsymbol{w}$. Threshold *freq* indicates the minimum number of trajectories for the overlapping regions to be considered as frequent routes for o. For each trajectory of o in $\boldsymbol{w}$, the algorithm identifies the links that it traverses and keeps a counter on the traversals. Then, it uses the collected counters to retrieve the links of the network that are frequently traveled by the moving object. The derivation of the frequent routes of object o in $\boldsymbol{w}$, from the corresponding links, is performed by examining the end points of the links and by joining together links that share end points. To identify the global frequent routes of an object, Algorithm 2 has to be executed for the different time intervals $\boldsymbol{w}$. Through the knowledge of the frequent routes of a moving object, the computation of its safe and unsafe routes, given a specific value of K as declared in the user profile, is straightforward.

In what follows, we investigate some issues that pertain to the automatic identification of the frequent (safe or unsafe) routes of each moving object o in the system.

4.2.1 Selecting the appropriate time intervals $\boldsymbol{w}$ Since the identification of the frequent routes of a moving object o depends on a set of time intervals $\boldsymbol{w}$, a first issue that needs further investigation regards the rationale behind the selection process for these intervals. By construction, ***w*** allows us to account for small time differences that are expected to be encountered in the regularly followed itineraries of a user. For example, a person may leave her house at 8:02am one morning to go to work, while the next day she leaves at 8:15am to follow the same itinerary. As is obvious, the system should be capable of identifying that the followed routes are the same (in a spatiotemporal sense) even if they do not perfectly coincide in time. Furthermore, depending on the existing network traffic conditions, a user may need a different amount of time for the same itinerary from one day to another.

Generally speaking, we consider that the selection of the appropriate size of ***w*** is an application-specific task. However, the size of ***w*** is expected to be smaller when user movement is more probable (e.g., at rush hours) and larger when user immobility is expected (e.g., during the late night hours). By using this convention, a reasonable amount of frequent routes is expected to be derived, leading to the best possible protection of the privacy of the user. Finally, it is also possible to have the different time intervals (i.e., ***w***'s) be automatically adjusted by the system based on the movement history of each user (i.e., recorded as part of the user profile and subsequently used to derive the frequent routes for this particular user). This will allow for accounting for different movement behaviors of distinct users in the system.

4.2.2 Addressing (computational) cost-related issues

The computation of the frequent routes of a user is a costly operation, particularly due to the expected large number of the trajectories that have to be taken into consideration. However, there are two key issues that allow us to proceed towards this direction:

- The computation can proceed offline through the use of a dedicated server and thus do not affect the online operation of the system. When computed, the new frequent routes will replace the old ones that are currently associated with the user profile.
- The behavior of a user is not expected to radically change from day to day, and even if it does change, it will still take some time until the old frequent routes become invalid (i.e., until the new routes of the user become frequent). As a result, the frequent routes of a user do not have to be constantly updated but only once in a while, based on the used *freq* threshold.

4.3 Phase III: Trajectory K-anonymity

The third phase in the process of privacy preservation is the provision of the trajectory K–anonymity. If K–anonymity is successfully provided to the user, then this is also the last phase of the process. Algorithm 2 computed the list of frequent routes *FR* for each moving object o. In order for the privacy preservation mechanism to adequately protect the privacy of a requester, the system should be capable of deciding when a user request matches a frequent route of the user and, subsequently, whether this route is safe or unsafe. In what follows, we concentrate on the first part. Regarding the second part, the classification of the frequent routes of a user into safe and unsafe is a matter of building an efficient lookup table that maintains this information.

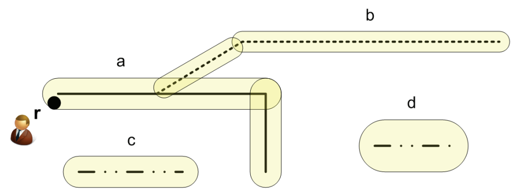


Figure 5: Matching requests to frequent routes.

Consider Fig. 5 that depicts a user o initiating a request r for an LBS. Assume that the temporal coordinate of the request matches four of the frequent routes of the user (i.e., 'a', 'b', 'c' and 'd' as shown in Fig. 5), each of which is buffered based on the width of the corresponding road. Moreover, as is shown in Fig. 5, some of these routes share common regions (e.g., 'a' and 'b'), while others are disjoint (e.g., 'c' and 'd'). The system should be capable of identifying that this request was made from within the regions of a frequent route of user o and proceed to provide K*–frequent* anonymity (if 'a' is a safe route) or K*–present* anonymity (if 'a' is an unsafe route). The matching part of the trajectory anonymization process proceeds in two steps: The first step examines the location of the request and identifies if it lies inside any of the frequent routes of the user. If this is the case, then the system takes the appropriate actions depending on whether the route is safe or unsafe. The second step takes place at each subsequent location update transmitted by the mobile device of the user and for as long as the user remains within a frequent route. If the user leaves his or her frequent routes, then the system offers K*–present* anonymity and continues to check the subsequent locations to detect whether the user has entered a frequent route. This process continues until the service is provided to the user.

Algorithm 3 provides the details of our implementation. In the first run, the algorithm isolates from the set of all the frequent routes of user o the ones that are time-consistent with respect to the time of request t. Then, we proceed to identify if the location (x, y) of the user at the time of request lies inside any of these geometries. The matching geometries, if any, are maintained in a list $\mathcal{M}$. Notation `geom`(*FR*) (lines 4, 9 and 13 of Algorithm 3) is used to demonstrate that only the spatial component (i.e., the geometry) of the frequent route is currently examined. Thus,

Algorithm 3 Match of requests to frequent routes.

Input: Current user request R_c, user's frequent routes *FR*, indicator *frun* of whether this is the first or an intermediate user request for this service and (if not the first request) the matched route of the last user request *FR*'.
Output: Boolean value denoting whether the match was successful or unsuccessful.

```
 1: function MATCHREQUEST($R_c$, FR, frun, <in-out> FR')
 2:     if frun = true then                                   ▷ this is the first request
 3:         FR ← $\{(F, w) \in FR | t \in w\}$              ▷ frequent routes of the user in w
 4:         $\mathcal{M}$ ← GET_GEOM_INSIDE($R_c|_{x,y}$, geom(FR))
 5:         if $\mathcal{M}$ = null then
 6:             return false                           ▷ request made from infrequent route
 7:         else
 8:             declare List FR' ← ∅
 9:             FR' ← GET_ANYINTERACT($\mathcal{M}$, geom(FR))        ▷ identify frequent
    routes in FR that are not disjoint to the matched route $\mathcal{M}$
10:             return true                            ▷ request made from a frequent route
11:         end if
12:     else                                                        ▷ not the first run
13:         $\mathcal{M}$ ← GET_GEOM_INSIDE($R_c|_{x,y}$, geom(FR'))
14:         if $\mathcal{M}$ = null then
15:             frun ← true                                   ▷ initialize for the next run
16:             return false                                 ▷ user left the frequent route
17:         else
18:             return true                               ▷ user is still in a frequent route
19:         end if
20:     end if
21: end function
```

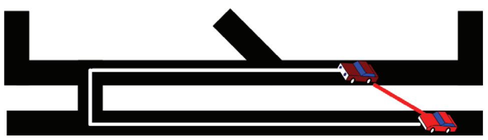


Figure 6: Euclidian vs network distance for the computation of the participants in the anonymity set.

the temporal component is ignored. In the event that the request was initiated at a location outside the frequent routes in *FR*, the user is provided with K–*present* anonymity and the algorithm is re-executed in the next location update (unless the request has been serviced in the meanwhile). Otherwise, the system identifies all the geometries that are not disjoint to $\mathcal{M}$ and stores them in a list *FR*' that is used in subsequent location updates regarding the same service. List *FR*' is used to store the frequent route, where the request was initiated, along with all the related (non-disjoint) frequent routes from *FR*. Considering the scenario presented in Fig. 5, *FR*' will store routes 'a' and 'b'. This is due to the fact that since the user initiated the request when inside frequent route 'a', the only possible frequent routes that he or she can follow without traversing an infrequent route, are 'a' and 'b'. Given the fact that a user may have a substantial number of frequent routes in the considered time frame ***w***, the use of the routes in *FR*' typically alleviates the system from a lot of unnecessary work.

Algorithm 4 presents the way that trajectory K–anonymity is delivered to a user that sent a request R_c given the spatial threshold A_{min}. The goal of the algorithm is to identify the K–1 trajectories of the users, for the time interval ***w*** $= [t_{n-1}, t_n]$ that satisfy the appropriate anonymity re-

Algorithm 4 Offering of trajectory K–anonymity.

Input: User request R_c, value of K in K–anonymity, minimum generalization area A_{min}.
Output: The computed circular anonymity region $\mathcal{C}$ that satisfies A_{min}.

```
 1: function GENERALIZEREQUEST($R_c$, $K$, $A_{min}$)
 2:     declare List_of_Geom $\mathcal{AS}$ ← ∅              ▷ initialize the anonymity set
 3:     declare Geometry $\mathcal{C}$ ← ∅
 4:     $\mathcal{AS}$ ← GET_K_NN($R_c$, K–1, $\alpha$)        ▷ find the nearest neighbors of the
    requester
 5:     $C_K$ ← GET_CENTROID($\mathcal{AS}$, w)                ▷ centroid of the routes in w
 6:     $R_K$ ← GET_DISTANCE($C_K$, K–th NN)     ▷ compute the distance from the
    center to the furthest neighbor
 7:     if $R_K < \sqrt{\frac{A_{min}}{\pi}}$ then
 8:         $R_K \leftarrow \sqrt{\frac{A_{min}}{\pi}}$ ▷ ensure that the distance fulfils the $A_{min}$ requirement
 9:     end if
10:     $\mathcal{C}$ ← CREATE_CIRCLE($C_K$, $R_K$)        ▷ create the anonymity region $\mathcal{C}$
11:     return $\mathcal{C}$
12: end function
```

quirements and to return the spatiotemporal region that covers all the K users (including the requester). An important observation is that a trajectory does not need to lie completely inside the considered spatial region, for its user to be encountered in the anonymity set of o. Rather, even if a small part of the trajectory is inside the spatial region at some time in ***w***, the corresponding user is counted in the anonymity set. Another important issue relates to the calculation of the distances between the point of request and the various trajectories in the system. In our implementation, the distance between the requester and a trajectory is captured as the minimum distance between the point of request and the nearest point of the trajectory to the requester. This means that distances are not computed based on the underlying network of movement, a decision that is justified in Fig. 6.

Algorithm 4 identifies the K–1 routes that are nearest to the route of the requester in ***w*** and stores them in a list $\mathcal{AS}$. The α parameter in the computation of the K nearest neighbors (including the requester) indicates the type of anonymity that is sought (K–*present* vs K–*frequent*) to allow the algorithm to search among the corresponding users. Having identified the participants of the anonymity set, the algorithm computes the centroid of the corresponding routes (with respect to ***w***) in $\mathcal{AS}$. This point C_K will be the center of the anonymity region of requester r. Following that, the algorithm computes the distance R_K of C_K to the most distant route in $\mathcal{AS}$. The final step of Algorithm 4 is to compute the circle $\mathcal{C}$ centered at C_K and having a radius R_K. This circle corresponds to the anonymity region for user r. To ensure that the anonymity region satisfies the A_{min} privacy requirement, the computed radius of the circle is tested against the minimum acceptable radius $\sqrt{A_{min}/\pi}$ and is augmented if necessary. Figure 7 presents the way Algorithm 4 offers 9-anonymity to user r. The center of the circular anonymity region corresponds to the centroid of the users. Notice that by construction, all the K users in the anonymity region are equi-probable to be the senders of request r, since the cen-

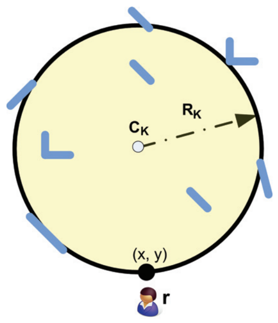


Figure 7: The computed region of K–anonymity.

troid of the routes in $\mathcal{AS}$ is uniquely defined.

In a service that can be completed within one location update, Algorithm 4 produces the anonymity region that replaces the space/time coordinates of the original request. On the other hand, if multiple location updates are necessary for the completion of the service, then the following strategy is applied. First, Algorithm 4 is executed and the area of request is generalized to satisfy K–anonymity. The K–1 subjects of the computed anonymity set are maintained by the trusted server for as long as the current LBS withstands completion. At each subsequent location update that needs to be transmitted to the service providers, the trusted server adjusts the time coordinate of the request and applies Algorithm 4 (lines 5–10) to generate a region of anonymity that contains (possibly among others) the original K subjects. This new region of anonymity, provided that it satisfies the A_{max} threshold, blocks the sequential tracking attack and offers privacy to the requester. However, if the A_{max} threshold is surpassed, the privacy of the user is under threat and a service delaying or denial mechanism has to take over.

4.4 Phase IV: Service delaying or denial There are circumstances when the proposed approach for the offering of trajectory K–anonymity to the requesters of LBSs may fail. This situation may occur for several reasons, such as an exceptionally high value of K defined in the profile of the requester or tight spatiotemporal generalization constraints (A_{max}, $\boldsymbol{w}$) required for the provision of the service. In all such cases, an approach has to be employed to effectively protect the identity of the requester. The proposed approach introduces a small delay in the servicing of the request, in the hope that in the meanwhile (i) more users will approach the requester, thus become part of his or her anonymity set, and/or (ii) the requester will move to a more populated region. Specifically, we defer the servicing of the request for a pre-specified time period and then re-execute Algorithm 4 for the provision of trajectory K–anonymity. If the algorithm fails again, then we protect the privacy of the user by denying the servicing of his or her request.

5 System implementation

In this section, we investigate some details that pertain to the implementation of the presented algorithms as part of the trusted server. Our implementation relies on a spatial database engine that stores all the necessary types of spatial (network of user movement) and spatiotemporal data (movement history, trajectories, frequent routes) and efficiently performs a set of spatial and temporal calculations for the provision of trajectory anonymization. A table stores all the users in the system along with references to their profiles and histories of movement. The profile of a user corresponds to a set of triples $<$ *sid*, K, A_{min} $>$ and is kept in a (spatial) table. Furthermore, tight to each profile is the set of frequent routes of the corresponding user. These, are kept in a (spatial) table consisting of tuples in the form of $<$ *routeID*, *geometry*, *time-interval* $>$, where the geometry depicts the traveled route through the network as a collection of the links that were traversed by the user, and the time-interval corresponds to the (unanchored) time that this route is typically followed by the user. For each frequent route, a table stores the values of K for which the route is safe or unsafe for the user. The user history of movement is kept in two separate tables: the table of the complete history of movement and the table of the user trajectories. Both tables store the data as a collection of geometries related to the actual segments, their start-time and end-time. A *sequenceID* is tight to each segment to denote its position in the geometry and to allow for the unification of the segments to build larger parts of a trajectory. Besides the user-related data, the spatial DBMS also stores the underlying network topology $\mathcal{N}$ of user movement through a set of structures provided by the DBMS. Finally, all the geometries in the spatial tables are indexed using R–trees and the metadata of the network are updated to include the appropriate costs for the various calculations.

The implementation of the proposed model is based on the Oracle DBMS and Java, and makes use of the Oracle Java API to allow their communication. The choice of Oracle was made primarily due to its inherent support for spatial networks, a support that is currently missing from other DBMSs. On the other hand, Java allows for the maximum open-endness of our framework. As a first step in our implementation, we use a network aware trajectory data generator that generates a set of trajectories (as a series of location updates) which are consistent to a given network. In sequel, both this data and the underlying network are transformed into the appropriate formats and are stored in Oracle. Finally, through the supported spatial functions of the DBMS our privacy model reconstructs the user history, computes the frequent routes and provides trajectory K–anonymity. Although our implementation was based on the spatial engine of Oracle, the proposed framework can be easily migrated (with the necessary adjustments) to any other spatial DBMS. To support this claim, Table 1 summarizes the spatial func-

Table 1: Mapping the spatial functions to their equivalents in the Oracle and IBM DB2 DBMSs.

Spatial Function	Oracle Spatial DBMS	DB2 Spatial DBMS
GET_m_SHORTEST_ROUTES	NETWORK_MANAGER.ALL_PATHS(m, constraints)	*unsupported*
GET_NEAREST_NEIGHBOR	NETWORK_MANAGER.NEAREST_NEIGHBORS(1)	*unsupported*
GET_OVERLAPS	SDO_OVERLAPS	DB2GSE.ST_Overlaps
GET_GEOM_INSIDE	SDO_INSIDE	DB2GSE.ST_Contains
GET_ANYINTERACT	SDO_ANYINTERACT	DB2GSE.ST_Relate
GET_K_NN_DISTANCE	SDO_NN (k) & SDO_WITHIN_DISTANCE(dist)	DB2GSE.ST_Distance
CREATE_{CIRCLE, SQUARE}	SDO_GEOMETRY	DB2GSE.ST_Geometry
GET_CENTROID	SDO_GEOM.SDO_CENTROID	DB2GSE.ST_Centroid

tions that were used in the provided algorithms and presents their equivalents for two major spatial DBMSs, namely Oracle and IBM DB2. Similar functions can be found in other DBMS with spatial capabilities, as well as functions needed for the computation of buffers, the identification of the length of roads or road parts, and the identification of geometries that are within a distance from an origin. This functionality is delivered in Oracle through functions SDO_BUFFER, SDO_LENGTH and SDO_WITHIN_DISTANCE.

Table 2: The characteristics of the datasets.

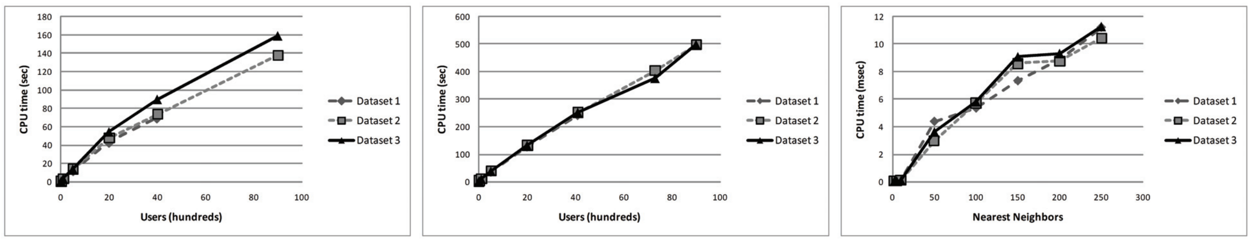
Parameter	Dataset 1	Dataset 2	Dataset 3
#objects	4,100	9,200	96,500
#classes	2	6	6
max time	2,000	3,000	8,000
report probability	1	0.9	0.7

6 Experimental evaluation

The proposed algorithms were implemented using Java and Oracle on Windows XP on a 3.2 Ghz Intel Pentium D processor equipped with 4GB of main memory. To evaluate the algorithms, we used Brinkhoff's network aware generator of moving objects [5] and generated three datasets of trajectories based on the road network of the Oldenburg city. Brinkhoff's generator is commonly used in most of the papers related to spatiotemporal datasets, as it models real world networks with synthetic data. The generated datasets experience different characteristics in terms of the number of moving objects and their classes, the maximum time of movement, and the probability of an object to report its position at each time. Table 2 summarizes their properties. The number of classes denotes the different types of moving objects (e.g., cars, trailers, motorbikes) that are considered in the generated data. Each moving object is assigned to a unique class that defines its maximum allowable speed in the network. Furthermore, the time of generation, the duration of existence and the time of disappearance of a moving object are different for the various objects. In conformance to Brinkhoff's generator, we use an integer type as the unit time instance *un* and set the whole time period from 0 to the corresponding maximum time as reported in Table 2. Note that since our framework is unique in the utilization of the underlying network for the offering of online trajectory anonymity, no comparisons have been made with other approaches.

In Brinkoff's generator, each object is routed through the links of the network to move from its origin to its final destination. The use of a routing algorithm for the generation of movement has as a consequence that no subtrajectory of an object is repeated and thereof no frequent routes exist. To solve this issue, for each moving object, we randomly selected a set of consecutive location updates (creating a temporal interval ***w***) and produced 3 new trajectories of this object that contained only these updates. Thus, we achieve to artificially generate frequent routes for the various users. The resulting datasets were used in the experiments.

We conducted three sets of experiments. The first set, presented in Fig. 8, tests the scalability of the proposed algorithms. Specifically, Fig. 8(a) shows the time that was needed to update the last segment of a user's history of movement (for a number of users), given knowledge of the previous location of the user. The approach first checks if the new location update lies in the same link of the network as the last known, and otherwise executes Algorithm 1. As one can observe, the runtime for dataset 3 is higher compared to the other two datasets, since, due to the irregular location updates, Algorithm 1 is executed more often. Moreover, since we have no knowledge regarding the time needed for the traversal of each link of the network, Algorithm 1 identifies the shortest route with respect to the distance that needs to be traveled by the object. Figure 8(b) shows the time needed to identify the frequent routes of each user and generate the corresponding lookup table. In this experiment, the ***w*** parameter is adjusted based on the values that were selected on trajectory addition stage, described earlier. The value of *freq* was set to 3. Figure 8(c) presents the time needed to compute the nearest neighbors to a requester, which is the primary operation of K–anonymization. As one can observe, the figure demonstrates the runtime that is needed for the computation of the anonymity set in K*–present* anonymity since it selects



(a) Reconstruction of user movement (b) Identification of frequent routes (c) Provision of K-anonymity

Figure 8: Scalability of the algorithms in the network aware model.

all the nearest neighbors of the requester. However, the runtime of K–*frequent* anonymity (for the same value of K) can be approximated through the graph by considering that more neighbors than K have to be found near the requester. For this experiment, a hundred requests were randomly generated, each by selecting a location update from the history of movement of a user, considering this as the point of request and setting ***w*** to have an extent of 10 location updates[2]. The reported times are averages over 100 runs.

The second and third sets of experiments (shown in Fig. 9 and Fig. 10, respectively) illustrate the success ratio of the proposed K–anonymity approaches, when applied on continuous queries where the servicing of the request requires 3 location updates. Each experiment involves a set of spatial (denoted as the maximum radius of the K–anonymity region) and temporal (denoted as the maximum number of time units *un* prior to the current time) constraints, and a value of K in 10–100. In all experiments, we set the A_{min} parameter such that the anonymity set is guaranteed to cover a circular area of radius equal to 50 meters, away from the requester. In both cases of trajectory anonymity, failure corresponds to the lack of identifying the necessary neighbors of the requester throughout the period of the servicing of the request. The reported ratios are an average over 100 runs coming from randomly selected users and spatiotemporal points in their trajectories.

7 Conclusions

In this paper, we presented a privacy model for the offering of personalized trajectory K–anonymity by utilizing an underlying network of user movement. Through a series of phases, our proposed model reconstructs the user movement, identifies safe and unsafe routes for each user, filters incoming user requests and offers the most suitable among two variants of trajectory K–anonymity, namely K–*present* (weak) and K–*frequent* (strong) anonymity. In the event that anonymity fails to be offered to the user, an alternative approach is employed to ensure that the user is still protected. Through experiments, we demonstrated the effectiveness of our model towards protecting any portion of the user trajectory that is under threat.

[2]Please notice that this ***w*** refers to the extent of the temporal cloaking that is offered by the K–anonymity algorithm and is not related to the time interval that is used for the identification of the frequent routes of a user.

Acknowledgments

Aris Gkoulalas-Divanis and Vassilios S. Verykios are supported in part by the EU project GeoPKDD (IST–6FP–014915). Mohamed Mokbel's research is supported in part by the National Science Foundation under Grants IIS–0811998, IIS–0811935, and CNS–0708604.

References

[1] O. Abul, M. Atzori, F. Bonchi, and F. Giannotti. Hiding sensitive trajectory patterns. In *Proceedings of the 7th IEEE International Conference on Data Mining Workshops*, pages 693–698, 2007.

[2] O. Abul, F. Bonchi, and M. Nanni. Never walk alone: Uncertainty for anonymity in moving objects databases. In *Proceedings of the 24th IEEE International Conference on Data Engineering*, pages 376–385, 2008.

[3] C. Bettini, X. S. Wang, and S. Jajodia. Protecting privacy against location-based personal identification. In *Proceedings of the 2nd VLDB Workshop on Secure Data Management*, pages 185–199, 2005.

[4] S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk. On map-matching vehicle tracking data. In *Proceedings of the 31st International Conference on Very Large Data Bases*, pages 853–864, 2005.

[5] T. Brinkhoff. A framework for generating network-based moving objects. *Geoinformatica*, 6(2):153–180, 2002.

[6] R. Cheng, Y. Zhang, E. Bertino, and S. Prabhakar. Preserving user location privacy in mobile data management infrastructures. In *Proceedings of the 6th Privacy Enhancing Technology Workshop*, 2006.

[7] G. Ghinita, P. Kalnis, and S. Skiadopoulos. Mobihide: A mobilea peer-to-peer system for anonymous location-based queries. In *Proceedings of the 10th International Symposium on Advances in Spatial and Temporal Databases*, pages 221–238, 2007.

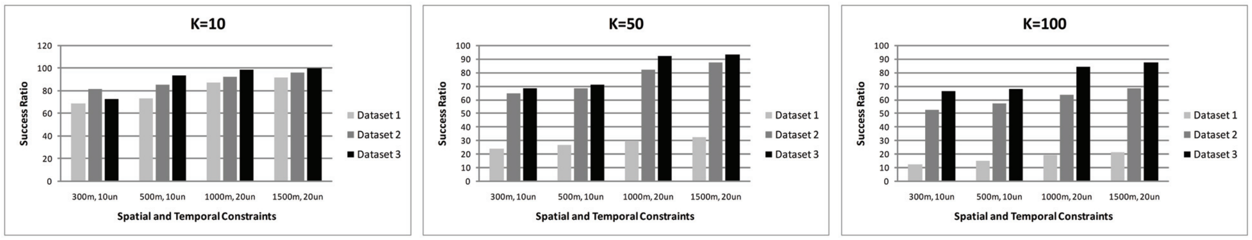


Figure 9: Success ratios for 10, 50 and $100-$*present* trajectory anonymity on continuous queries.

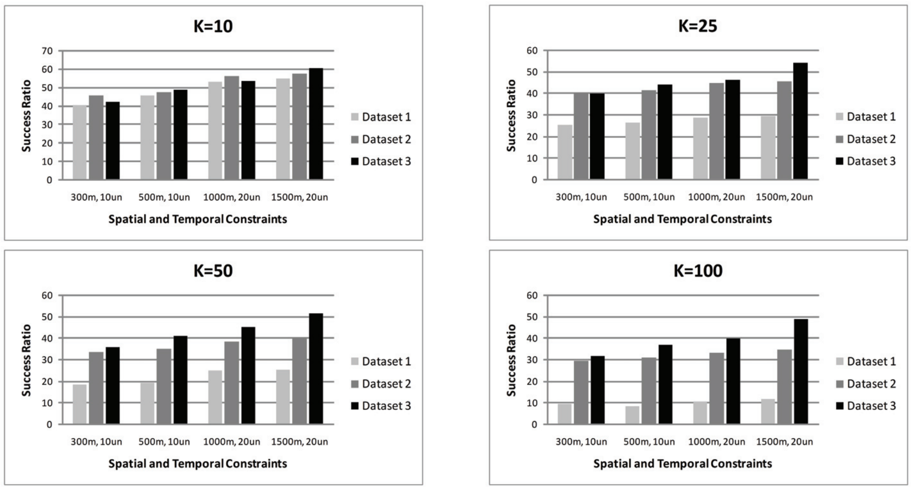


Figure 10: Success ratios for 10, 25, 50 and $100-$*frequent* trajectory anonymity on continuous queries.

[8] G. Ghinita, P. Kalnis, and S. Skiadopoulos. Prive: anonymous location-based queries in distributed mobile systems. In *Proceedings of the 16th International Conference on World Wide Web*, pages 371–380, 2007.

[9] F. Giannotti, M. Nanni, F. Pinelli, and D. Pedreschi. Trajectory pattern mining. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 330–339, 2007.

[10] M. Gruteser and D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *Proceedings of the 1st International Conference on Mobile systems, Applications and Services*, pages 31–42, 2003.

[11] M. Gruteser and X. Liu. Protecting privacy in continuous location-tracking applications. *IEEE Security and Privacy Magazine*, 2(2):28–34, 2004.

[12] P. Kalnis, G. Ghinita, K. Mouratidis, and D. Papadias. Preventing location-based identity inference in anonymous spatial queries. *IEEE Transactions on Knowledge and Data Engineering*, 19(12):1719–1733, 2007.

[13] A. Machanavajjhala, D. Kifer, J. Abowd, J. Gehrke, and L. Vilhuber. Privacy: Theory meets practice on the map. In *Proceedings of the 24th IEEE International Conference on Data Engineering*, pages 277–286, 2008.

[14] F. Marchal, J. Hackney, and K. W. Axhausen. Efficient map matching of large global positioning system data sets: Tests on speed-monitoring experiment in zurich. *Transportation Research Record*, 1935:93–100, 2006.

[15] N. Meratnia and R. A. de By. Trajectory representation in location-based services: Problems and solution. In *Proceedings of the 4th International Conference on Web Information Systems Engineering Workshops*, pages 18–24, 2003.

[16] M. F. Mokbel, C.-Y. Chow, and W. G. Aref. The new Casper: Query processing for location services without compromising privacy. In *Proceedings of the 32nd International Conference on Very Large Data Bases*, pages 763–774, 2006.

[17] P. Samarati. Protecting respondents' identities in microdata release. *IEEE Transactions on Knowledge and Data Engineering*, 13(6):1010–1027, 2001.

[18] L. Sweeney. *K*-anonymity: A model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5):557–570, 2002.

[19] M. Terrovitis and N. Mamoulis. Privacy preservation in the publication of trajectories. In *Proceedings of the 9th IEEE International Conference on Mobile Data Management*, pages 65–72, 2008.

[20] G. Trajcevski, O. Wolfson, K. Hinrichs, and S. Chamberlain. Managing uncertainty in moving objects databases. *ACM Transactions on Database Systems*, 29(3):463–507, 2004.

[21] H. Yin and O. Wolfson. A weight-based map matching method in moving objects databases. In *Proceedings of the 16th International Conference on Scientific and Statistical Database Management*, pages 437–446, 2004.