Utility-Aware Synthesis of Differentially Private and Attack-Resilient Location Traces

Mehmet Emre Gursoy, Ling Liu, Stacey Truex, Lei Yu, and Wenqi Wei
College of Computing, Georgia Institute of Technology

ABSTRACT

As mobile devices and location-based services become increasingly ubiquitous, the privacy of mobile users' location traces continues to be a major concern. Traditional privacy solutions rely on perturbing each position in a user's trace and replacing it with a fake location. However, recent studies have shown that such point-based perturbation of locations is susceptible to inference attacks and suffers from serious utility losses, because it disregards the moving trajectory and continuity in full location traces.

In this paper, we argue that privacy-preserving synthesis of *complete* location traces can be an effective solution to this problem. We present AdaTrace, a scalable location trace synthesizer with three novel features: provable statistical privacy, deterministic attack resilience, and strong utility preservation. AdaTrace builds a generative model from a given set of real traces through a four-phase synthesis process consisting of feature extraction, synopsis learning, privacy and utility preserving noise injection, and generation of differentially private synthetic location traces. The output traces crafted by AdaTrace preserve utility-critical information existing in real traces, and are robust against known location trace attacks. We validate the effectiveness of AdaTrace by comparing it with three state of the art approaches (ngram, DPT, and SGLT) using real location trace datasets (Geolife and Taxi) as well as a simulated dataset of 50,000 vehicles in Oldenburg, Germany. AdaTrace offers up to 3-fold improvement in trajectory utility, and is orders of magnitude faster than previous work, while preserving differential privacy and attack resilience.

CCS CONCEPTS

• **Security and privacy** → *Privacy-preserving protocols*; *Database and storage security*;

KEYWORDS

data privacy; mobile computing; location privacy

ACM Reference Format:
Mehmet Emre Gursoy, Ling Liu, Stacey Truex, Lei Yu, and Wenqi Wei. 2018. Utility-Aware Synthesis of Differentially Private and Attack-Resilient Location Traces. In *2018 ACM SIGSAC Conference on Computer and Communications Security (CCS '18), October 15–19, 2018, Toronto, ON, Canada.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3243734.3243741

CCS '18, October 15–19, 2018, Toronto, ON, Canada

ACM ISBN 978-1-4503-5693-0/18/10…$15.00
https://doi.org/10.1145/3243734.3243741

1 INTRODUCTION

A growing number of location-based services and applications, such as those offered by Google, Uber, Lyft, and Waze, continue to collect users' location traces in order to learn about their spatio-temporal movement patterns and offer life enriching, real-time experiences through location-based convenience and entertainment [1, 3, 38]. On the other hand, the sensitive nature of location trace data raises legitimate privacy concerns. Unauthorized exposure of users' private location traces may disclose their travel records, home and work locations, frequent meeting points, or visits to sensitive locations such as hospitals, health clinics, and religious events.

Traditional location privacy protection techniques have mostly focused on point-based location privacy, which is often achieved by perturbing or obfuscating each location point in a user's trace using a cloaked region or a fake location, with the goal of ensuring location k-anonymity or geo-indistinguishability [5, 8, 23, 37, 39, 46, 47, 54]. However, these point-based privacy mechanisms are insufficient for protecting the privacy of users' *trajectories*, i.e., spatially correlated, temporal sequences of locations. Several studies show that independent perturbation of each point-based location in a trajectory suffers from fatal shortcomings, including susceptibility to reverse engineering and inference attacks [7, 51]. In these attacks, adversaries observe a sequence of perturbed locations to infer a movement pattern and then link specific movement patterns with specific users. Such perturbations also suffer from acute spatial utility losses [10], and are vulnerable to known location trace attacks.

The aforementioned shortcomings motivated recent interest in synthesizing privacy-preserving, complete location traces [7, 11, 26]. These recent trace synthesis mechanisms focus on either differential privacy [11, 26] or plausible deniability [7], and while they offer desirable theoretical guarantees, they are limited by the extent of their chosen privacy definitions. For example, Dwork showed the impossibility to achieve absolute disclosure prevention [18]. A resulting limitation is that differential privacy cannot bound *all* prior and posterior knowledge distributions of an adversary, which sparked interest in augmenting a Bayesian form of privacy with prior-to-posterior comparison [28, 31, 32]. This Bayesian property is clearly useful for location privacy, e.g., even if we cannot bound all possible knowledge that is disclosed to an adversary, a Bayesian formulation can be used to limit disclosure specifically in sensitive zones such as hospitals or schools. Another example is regarding outliers – it is well-known that differential privacy requires large noise amounts to mask the existence of outliers. Location traces are typically high-dimensional and contain diverse behavior. Hence, it is worthy to seek alternative attack resilience notions to combat privacy leakages concerning numerous kinds of outlier traces.

We argue that a practical solution to synthesizing privacy-preserving location traces should achieve three goals simultaneously: (i) robust, well-defined statistical privacy, (ii) deterministic resilience against location privacy attacks, and (iii) strong preservation of trajectory utility and authenticity. We present AdaTrace, a scalable and quantitative framework that provides utility-aware synthesis of differentially private and attack-resilient location traces. AdaTrace builds a *generative* model over a dataset of real location traces by performing feature extraction, noise injection, and feature synthesis throughout four phases. In each phase, it first extracts utility-critical features and then allocates an adequate differential privacy budget to inject sufficient perturbation (noise), while maintaining attack resilience and preserving desired statistical and spatial utility. We guarantee that the synthetic trajectory generation process is differentially private, i.e., the generation of synthetic traces is not strongly dependent on any specific trace in the real trace dataset. This ensures unlinkability between an output trace and any real trace (input), thus thwarting identity and record linkage attacks. As such, although the location traces crafted by AdaTrace bear a statistical resemblance to real traces, they do not leak information about any individual that participates in the synthesis process.

We also ensure that the traces generated by AdaTrace are robust against three attacks: Bayesian inference, partial sniffing, and outlier leakage. While the generation process is probabilistic to satisfy differential privacy, we enforce attack resilience by imposing deterministic constraints on the generated traces. Finally, by maintaining resemblance to real traces, the crafted traces preserve many useful statistical features that are in common with real traces, and therefore they can be effectively used in geo-spatial queries and geo-spatial data mining tasks.

Contributions. The design of AdaTrace is novel in three aspects. First, we identify three privacy threats that are relevant and common in trajectory data, and are neither addressed by differential privacy nor by other existing trajectory generators. We formalize these threats as Bayesian inference, partial sniffing, and outlier leakage attacks, and propose defenses for mitigation. We show that in many cases, our defenses can be realized with little or no aggregate utility loss, demonstrating their effectiveness. Second, we combine differential privacy with attack resilience to offer a two-layer privacy approach. This allows us to integrate statistical, algorithmic privacy with output privacy, enabling deterministic attack resilience in a probabilistic trace synthesis setting. Third, we design a utility-aware trace synthesizer by analyzing and categorizing various types of location trace utility and by devising noise-resilient utility extraction and preservation techniques.

We validate the effectiveness of AdaTrace by comparing it with three representative existing approaches (ngram [11], DPT [26], and SGLT [7]) using both real location trace datasets (Geolife [56] and Taxi [38]), as well as a dataset simulated by Brinkhoff simulator [9] of 50,000 vehicles in Oldenburg, Germany. We empirically measure how well each utility type is preserved, and our results show that AdaTrace offers up to 3-fold improvement in trajectory utility while preserving both differential privacy and attack resilience. AdaTrace is also computationally more efficient, as it is on average 1.5x faster than DPT, 8x faster than ngram, and 1000x faster than SGLT.

2 RELATED WORK

Location Perturbation. Location privacy has been studied for over a decade. Most existing solutions employ different location perturbation techniques to compute a k-anonymous or geo-indistinguishable location cloaking region around a true location coordinate in order to hide the true location in a crowd of fake locations [5, 8, 23, 37, 39, 46, 47, 54]. Location queries are then modified by adding noise to user's real location coordinate – they either use a low resolution version of the real location coordinate (such as the cloaked region), or a randomly chosen fake location within the cloaked region. Location queries protected using such randomly selected fake locations within a cloaked region tend to suffer from inference attacks with high attack success rate [7] due to the strong statistical correlation between the fake location and the user's true location.

Synthetic Location Generation. As opposed to location perturbation, some research has been performed to protect privacy by generating synthetic locations. This line of work is inspired by the security research on generating fake or synthetic records for privacy protection in web search, anonymous communication, and authentication systems [24, 27, 41]. In all scenarios, the main challenge is to generate synthetic data in a manner that resembles genuine user-produced data while offering practical privacy protection at the same time. In the context of location-based systems, some work has been on generating and injecting a synthetic point-based location within a user's trajectory [29, 46, 53]. A few recent projects studied the problem of generating synthetic trajectories [7, 11, 26]. ngram [11] and DPT [26] proposed to generate fake trajectories with differential privacy. Although offering strong statistical privacy guarantees, we argue that relying solely on differential privacy has two shortcomings. First, its probabilistic nature does not allow deterministic protection against targeted attacks such as those considered in our paper. Second, it places restrictions on the generation algorithm but not its output, and therefore sensitive inferences remain possible [13, 14, 21]. In contrast to ngram and DPT, SGLT [7] enforces *plausible deniability* to generate privacy-preserving fake traces. It first introduces trace similarity and intersection functions that map a fake trace to a real trace under similarity and intersection constraints. Then, it generates one fake trace using one real trace as its seed. If the fake trace satisfies plausible deniability, i.e., there exist k other real traces that can map to the fake trace, then it preserves privacy of the seed trace.

3 ADATRACE OVERVIEW

3.1 Problem Statement

Consider a database of real location traces (trajectories) collected from mobile travelers on the road, denoted by D_{real}. We want to build a generative model over D_{real} using feature extraction, noise injection, and synopsis formation while upholding statistical utility and attack resilience. We then employ the generative model to output a set of synthesized traces, denoted by D_{syn}. This probabilistic generative model should be differentially private such that removing any real trace from D_{real} has no significant impact on the outcome D_{syn}. In addition, the synthetic traces D_{syn} should collectively retain a high resemblance to the real traces D_{real}, so that D_{syn} has many useful statistical and spatial features in common

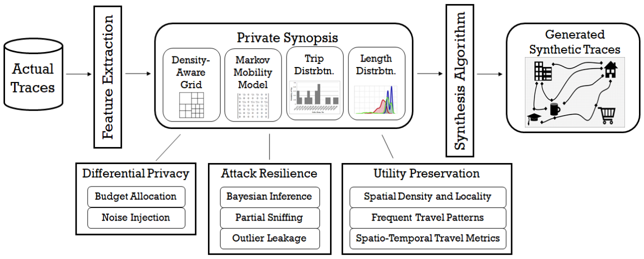

Figure 1: AdaTrace system architecture

with D_{real}. Finally, both the generative model and the synthetic traces D_{syn} should be robust against inference attacks, in order to strengthen privacy by maximizing attackers' probability of error in estimating the true behavior of users over time.

We design and develop AdaTrace, a scalable and utility-aware trajectory synthesizer with differential privacy and attack resilience. To demonstrate its feasibility and effectiveness, we compare AdaTrace with existing state of the art mechanisms on both real and simulated location trace datasets [9, 38, 56]. AdaTrace methodically unites statistical privacy (e.g., differential privacy) and syntactic privacy (e.g., attack resilience). It maximizes privacy by mitigating location inference attacks, and minimizes utility loss by extracting and upholding many useful types of statistical and spatial features of real location traces throughout each phase of synthesis.

Figure 1 illustrates the system architecture of AdaTrace. First, AdaTrace analyzes D_{real}, the database of real location traces, to extract four spatial and statistical features and establish them in its private synopsis. The features include a density-aware grid to capture the complex spatial density and locality distributions; a Markov mobility model to capture intra-trace mobility patterns; a trip distribution to learn the correlations between trips' pickup and destination locations; and a length distribution to capture travel distances and durations. Each of the four features are perturbed by controlled noise under differential privacy and attack resilience constraints. Although several mechanisms can be used to add noise during each feature extraction and perturbation phase, in AdaTrace we carefully choose the mechanisms such that: (i) we find a balance in the trade-off between privacy, attack resilience and utility, and (ii) we maximize utility under the same level of privacy and robustness.

3.2 Differential Privacy

We adapt the well-known notion of differential privacy to achieve statistical privacy in AdaTrace. We enforce differential privacy with respect to the complete location trace of an individual. It was recently argued that such use of the differential privacy notion can combat membership inference attacks relevant in machine learning as well as aggregate location statistics [42, 45].

For an actual database D_{real}, let $nbrs(D_{real})$ denote the set of databases neighboring D_{real} such that for all $D'_{real} \in nbrs(D_{real})$, it holds that $(D_{real} - D'_{real}) \cup (D'_{real} - D_{real}) = \{T\}$, where T denotes one user's trajectory. Further, let $\mathcal{A}$ be a probabilistic algorithm, ε be the privacy parameter, and $Range(\mathcal{A})$ denote the set of $\mathcal{A}$'s possible outcomes. $\mathcal{A}$ is said to be ε-differentially private [18], if for all D_{real} and $D'_{real} \in nbrs(D_{real})$, and for all possible outcomes $S \in Range(\mathcal{A})$:

$$Pr(\mathcal{A}(D_{real}) = S) \leq e^{\varepsilon} \times Pr(\mathcal{A}(D'_{real}) = S)$$

A special case of this definition is when $\mathcal{A}$ is a synthetic trajectory generation process and $Range(\mathcal{A})$=$\{D^1_{syn}, ..., D^n_{syn}\}$ is the set of all possible synthetic databases, yielding AdaTrace's differential privacy requirement. This requirement guarantees that the output of a private algorithm does not enable an adversary to distinguish, beyond a probability controlled by parameter ε, between two input databases D_{real} and D'_{real} that differ in one user's GPS trace. Hence, an adversary observing D_{syn} or an intermediate product of AdaTrace, including the features of the private synopsis, will not be able to infer the existence or content of a user's location data in D_{real} with strong confidence. Smaller ε yields tighter privacy [20].

We use two private building blocks in AdaTrace, namely the Laplace and Exponential mechanisms, for numeric and categorical queries respectively. Let f be a real-valued function $f : D_{real} \rightarrow \mathbb{R}^m$ that maps a database D_{real} into a fixed-size vector of m real numbers. The sensitivity of f, denoted Δf, is defined as the magnitude of the maximum L_1-norm change in the result of the function on all possible neighboring databases:

$$\Delta f := \max_{D_{real}, D'_{real}} ||f(D_{real}) - f(D'_{real})||$$

When answering a set of numeric (i.e., real-valued) queries, the Laplace mechanism retrieves the true answers of these queries, and then perturbs each answer by adding random noise scaled according to their sensitivity. That is, let $Lap(\alpha)$ denote a random variable sampled from the Laplace distribution with mean 0 and scale parameter α. The differentially private algorithm $\mathcal{A}$ answers f by [19]:

$$\mathcal{A}(f, D_{real}) = f(D_{real}) + (Y_1, ..., Y_m)$$

where Y_i are i.i.d. random variables drawn from $Lap(\Delta f / \varepsilon)$.

When the query is categorical instead of real-valued, the Exponential mechanism is used. It selects a discrete output r from a domain of outputs R in a private manner [34]. It employs a scoring function (i.e., quality criterion) q that associates each output $r \in R$ with a non-zero probability of being selected. Let $q(D_{real}, r)$ denote the quality of returning output r given database D_{real}, and let Δq be the sensitivity of q, defined similarly to Δf above. Then, the Exponential mechanism returns output $x \in R$ with probability equal to:

$$Pr(\mathcal{A}(q, D_{real}) = x) = \frac{e^{\frac{\varepsilon q(D_{real}, x)}{2\Delta q}}}{\sum_{r \in R} e^{\frac{\varepsilon q(D_{real}, r)}{2\Delta q}}}$$

The composition properties enable us to combine the building blocks and generate more sophisticated algorithms [35]. Because of the composition properties, ε is also called the *privacy budget*. Given a total budget ε, our goal is to create a sophisticated algorithm by cleverly combining the building blocks according to the composition properties. We employ three composition properties. First, for n algorithms $\mathcal{A}_1 \ldots \mathcal{A}_n$, each satisfying differential privacy with parameter $\varepsilon_1 \ldots \varepsilon_n$, the sequential execution of these algorithms on D_{real} consumes $\sum_{i=1}^{n} \varepsilon_i$ from the budget. Second, if the algorithms are executed on disjoint subsets of the database, the resulting execution consumes $\max(\varepsilon_i)$. Third, post-processing the output of a differentially private algorithm does not consume any budget or deteriorate privacy. For example, modifying the output of $\mathcal{A}_1$ without accessing D_{real}, and releasing the modified version still consumes ε_1.

3.3 Privacy Threat Model and Attacks

Differential privacy is regarded as the de facto standard for privacy-preserving data sharing due to its provable privacy guarantees. However, like any other security mechanism, the resilience of generic differential privacy against targeted, syntactic attacks has recently been criticized [13, 14, 21, 28]. We categorize these attacks broadly into three types of threats which are highly relevant in sharing sanitized location traces, but are beyond the scope of protection offered by differential privacy. We describe AdaTrace's threat model with respect to each attack and discuss why differential privacy is not sufficient to solve the attacks.

Bayesian Inference Threat. Certain geographic zones (regions) are sensitive by nature, e.g., hospitals, schools, health clinics. Let Z denote a privacy sensitive zone. An adversary may have a prior belief $\mathcal{B}(Z)$ regarding the users who visit these zones, e.g., where they live, work, or which other locations they frequently visit. We allow informed and uninformed priors, but by default address stronger adversaries with *informed priors*, e.g., knowledge obtained from publicly available information such as census records and population averages. For example, if 10% of the general population lives in a certain neighborhood, then $\mathcal{B}(Z)$ may assume that 10% of the users who visit the hospital also live in that neighborhood.

The adversary formulates a posterior belief after observing the synthesized trajectories, denoted by $\mathcal{B}(Z|D_{syn})$. If the difference between $\mathcal{B}(Z)$ and $\mathcal{B}(Z|D_{syn})$ is significant, we assert that there is a privacy leakage. For example, the adversary observes from D_{syn} that 50% of the hospital's visitors live in a certain neighborhood rather than the assumed prior of 10%, yielding the inference that there is a flu outbreak in that neighborhood. Differential privacy is not sufficient to prevent this threat, because despite its noise injection, priors and posteriors will be related. For example, if indeed 50% of the hospital's visitors come from a certain neighborhood in D_{real}, after Laplace noise we may have a 48% visiting rate from the same neighborhood in D_{syn}. This is still significantly higher than the 10% prior. A proper defense should ensure that the difference between $\mathcal{B}(Z)$ and $\mathcal{B}(Z|D_{syn})$ is sufficiently small, so that an adversary's inference power is crippled.

Partial (Subtrajectory) Sniffing Threat. Assume that an adversary can track users when they are in certain regions, such as grocery stores or airports. We call these regions *sniff regions.* Tracking in sniff regions can be enabled via surveillance cameras, biometric scanners, or simply by following users' location check-ins on a social network. The sniffed locations constitute a subtrajectory (i.e., portion) of the user's full trajectory. Armed with subtrajectory knowledge, if the adversary can infer which trajectory belongs to the user with high probability, the adversary can learn the remaining locations visited by the user outside the sniff region, such as home and work locations. Let u denote the sniffed user and $T_u \in D_{real}$ denote her trajectory. Even when D_{syn} is shared in place of D_{real}, an adversary can launch this attack by linking the subtrajectory $T_s \in D_{syn}$ which is geographically most similar to T_u in the sniff region. This inference process is illustrated in Figure 2. The threat is especially acute if T_s and T_u share points in one or more sensitive zone.

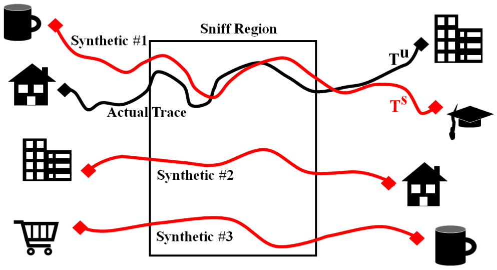

Figure 2: User's actual trace T_u drawn in black, synthetic traces passing through the sniff region drawn in red. Synthetic #1 is closest to T_u within the sniff region, hence it is labeled T_s.

Outlier Leakage Threat. A trajectory database may contain outliers that are unique in one or more ways such as the locations they visit, their start/destination points, and their time of travel. It is these outlier trajectories that are often hunted by adversaries. For example, while a $15 taxi ride in downtown Manhattan may not reveal much, a $125 trip from an isolated location to the airport leaks the identity and travel plans of a particular individual – note that this attack has actually been successfully exercised using NYC Taxi data [2].

We argue that relying solely on differential privacy cannot combat outliers for two reasons. First, since trajectories are often high-dimensional and unique [17, 48], there may exist skewed instances that are easily singled out and mapped to particular users with high confidence. Differential privacy needs to inject large noise amounts to hide such skewed instances. Second, even when large noise amounts are added, an outlier may still result from the probabilistic nature and algorithmic randomness of differential privacy. We should therefore place deterministic constraints to combat outliers – we combat an outlier trajectory T_{out}'s leakage threat by ensuring that it plausibly blends into a crowd of κ users' trajectories. This guarantees that T_{out} can at best be associated with a group of users (of size κ), and does not disclose the identity or secrets of any one user.

3.4 Utility Reference Model

The utility reference model used in AdaTrace is primarily designed based on the common utilities of location traces used in many geospatial data analysis and mining tasks, such as: users' frequency of visiting popular semantic locations, location entropy calculation, spatial decomposition, aggregate human mobility modeling (e.g., spatial, temporal and semantic mobility mining), and training predictive models for next location and destination inference. We identify the following three core utility notions as the prominent categories of trajectory utility in our reference model.

(1) Spatial density and locality distribution, e.g., for analyzing where users visit and their popular points of interest. This can have various commercial benefits, including choosing ideal geographic placement for advertisements or retail stores, spatial hotspot discovery, and recommendation in location-based social networks.

(2) Spatio-temporal travel metrics, e.g., correlations between users' trip start and destination locations, how long their trips take, etc. Since real datasets often consist of trips such as taxi and Uber rides, trajectories have well-defined pickup and destinations. The commercial benefits of preserving the correlations and related statistics include taxi service prediction and passenger demand analysis, as well as governance benefits such as emergency response planning.

(3) Frequent and representative travel patterns, e.g., commonly preferred routes of travel. Preserving this information helps discover associations or sequentiality between road segments, ultimately benefiting road network performance analysis, traffic flow control, and path recommendation in navigation services.

Although these utility categories are listed separately, they can be used in conjunction to enable more sophisticated geo-spatial data analysis and machine learning tasks, some of which are mentioned above. A natural question here is *what if instead of synthesizing privacy-preserving traces, we kept the original traces without modification but governed data analysts' access with a differentially private querying interface?* This interactive approach suffers from several drawbacks. First, it places an additional burden on the data owner to create, maintain, and enforce a privacy-preserving querying interface. Second, it jeopardizes the analysts' ability to apply off-the-shelf ML tools or libraries, since these tools assume availability of raw records and are not compatible with noisy query interfaces. Third, our generated traces can be used in arbitrary tasks, including those that were unforeseen at time of generation. In contrast, the querying interface is an ad hoc solution whose functionality must be enhanced each time a new type of query or analysis is desired. Due to these factors, our non-interactive trace generation strategy has much higher benefit and convenience for all parties.

4 SYNTHETIC TRAJECTORY GENERATION

In this section, we will describe the features in AdaTrace's private synopsis, as well as its trajectory synthesis algorithm. For each of the features, we will discuss how noise and perturbation is injected so that privacy is preserved. A central privacy parameter here is ε, the differential privacy budget. Since AdaTrace's synopsis consists of four features as shown in Figure 1, ε is divided into four sub-budgets ε_1 to ε_4, one for each feature, such that $\sum_{i=1}^{4} \varepsilon_i = \varepsilon$. Detailed discussion on the privacy and utility implications of the budget division is presented in Section 5.1.

4.1 Density-Aware Grid

Effective discretization of $\Omega(D_{real})$, the geographic location space of D_{real}, constitutes the first step towards preserving spatial densities. Without discretization, $\Omega(D_{real})$ is continuous and we have an unbounded number of possibilities to simulate a move from one location to another when generating a trajectory. To develop an efficient and accurate synthesis algorithm, we need to bound such transition possibilities by high utility choices. This motivates our design of a *grid* structure for space discretization. Although grids have been previously used in the location privacy literature [4, 6, 43], choosing an appropriate grid size and structure is not trivial under our privacy and utility constraints. If the grid is too coarse (e.g., 2x2), then each grid cell covers a large spatial area, and knowing that T visited a certain cell is uninformative. If the grid is too fine-grained (e.g., 50x50), we risk having many empty cells in which there are very few or no visits, and more additions of Laplace noise to each cell leads to higher inaccuracy as well as inefficiency.

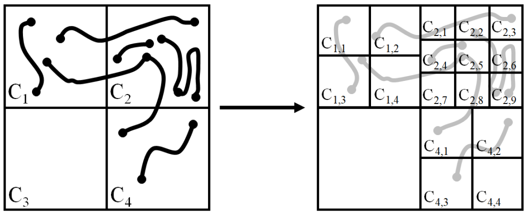

Figure 3: Two-step grid construction, top-level with $N = 2$ on the left and density-aware bottom-level on the right.

We therefore implement a grid with density-adaptive cell granularity [43]. That is, for low density regions in $\Omega(D_{real})$ we place large, coarse cells; whereas for high density regions we divide the region into smaller cells with finer granularity. The grid is built in two levels. In the top-level, we lay an $N \times N$ grid with uniform cells. Depending on the density of these cells, in the bottom-level, we subdivide each cell into varying sizes to reflect the density distribution of D_{real}. This allows us to *selectively* zoom into dense regions of $\Omega(D_{real})$ only when the need arises, which saves us from redundant computation and noise addition to sparse regions.

First, the grid is initialized with $N \times N$ identical cells, according to an input parameter N. This produces N^2 cells in the top-level, which we denote by $C_1, C_2, ..., C_{N^2}$. Then, we encode each trajectory in D_{real} as a list of cells, e.g., observe the trajectory $T : C_1 \rightarrow C_2 \rightarrow C_4$ in Figure 3. We denote by $|T|$ the number of cells T contains. Next, for each cell C_i, where $1 \leq i \leq N^2$, we count the *normalized* number of visits in that cell as follows:

$$g(D_{real}, C_i) = \sum_{T \in D_{real}} \frac{\text{\# of occurrences of } C_i \text{ in } T}{|T|}$$

We say that g is normalized since occurrences are divided by trajectory lengths. We need to find how much noise should be added to the query answers to satisfy differential privacy. In Appendix A we show that the sensitivity of the set of queries:

$$\boldsymbol{W} = \{g(D_{real}, C_1), g(D_{real}, C_2), ..., g(D_{real}, C_{N^2})\}$$

is $\Delta\boldsymbol{W} = 1$, therefore it suffices to add $Lap(1/\varepsilon_1)$ to each query answer to obtain the noisy answers denoted $\hat{g}(D_{real}, C_i)$. The reason why we perform normalization becomes clear from Appendix A – without division by $|T|$, we either have unbounded sensitivity $\Delta\boldsymbol{W}$ (implying unbounded noise) or we have to resort to alternative strategies to bound sensitivity (such as abruptly cutting trajectories or removing large portions therein), which are ill-suited for density measurement.

The final step in grid construction is to subdivide C_i using the noisy visit counts $\hat{g}(D_{real}, C_i)$. Each C_i is independently divided further into $M_i \times M_i$ cells, where M_i is proportional to $\hat{g}(D_{real}, C_i)$ and a grid constant. This achieves our goal of zooming into dense regions with large visit counts. For example, in Figure 3, we have densities $C_2 > C_1 \approx C_4 > C_3$, hence $M_2 = 3$, $M_1 = M_4 = 2$ and $M_3 = 1$. The grid constant acts as a balancing factor to determine the highest and lowest possible value of M, so that both M remains sensitive to changes in spatial density and the total number of cells resulting from this procedure is not excessively large.

4.2 Markov Chain Mobility Model

To generate high utility and realistic synthetic trajectories, we need to mimic actual intra-trajectory mobility. AdaTrace employs Markov chains for mobility modeling. A Markov chain of order r asserts that the next location in a trajectory depends on the previous r locations instead of all previous locations. Our grid-based discretization allows us to build a discrete state space Markov chain as follows. We map each cell in the adaptive grid to a state in our Markov chain. We assume each trajectory is represented as a time-ordered sequence of cells, and denote by $T[\text{j}]$ the jth entry in T. We find the transition probability of T to a next cell C_{next} having observed its previous n locations $T[1]T[2]\dots T[\text{n}]$:

$$\begin{aligned} &Pr\Big(T[\text{n+1}] = C_{next} \mid T[1]\dots T[\text{n}]\Big) \\ &\quad = Pr\Big(T[\text{n+1}] = C_{next} \mid T[\text{n-r+1}]T[\text{n-r+2}]\dots T[\text{n}]\Big) \end{aligned}$$

which boils down to the ratio of the number of sequences in T containing $T[\text{n-r+1}]T[\text{n-r+2}]\dots T[\text{n}]C_{next}$ divided by the sequences containing the prefix $T[\text{n-r+1}]T[\text{n-r+2}]\dots T[\text{n}]$. We say that the trajectory-specific mobility model, $\Pi(T)$, is the collection of such probabilities $Pr\big(T[\text{n+1}] \mid T[1]\dots T[\text{n}]\big)$. The trajectory-specific model captures the mobility of a single user in D_{real}. An aggregate mobility model for the whole D_{real} can be found by averaging the individual mobility models of each user. We slightly abuse notation and write $\Pi(D_{real})$ to denote the aggregate mobility model.

Similar to noise addition during grid construction, the aggregate model $\Pi(D_{real})$ is also perturbed with Laplace noise to satisfy differential privacy. We add noise to the Markov probabilities. Since Markov probabilities are computed using a ratio of sequence counts, and since these counts have sensitivity equal to 1, the required noise amount is limited. Hence, $\Pi(D_{real})$ can remain robust to noise.

4.3 Trip Distribution

Real life trace databases often consist of trips, such as taxi trips, Uber trips, home-work commutes, etc. The *trip distribution* aims to preserve the association between the start-end points of these trips when generating D_{syn}. This trip distribution is also useful from a technical sense to guide a random walk on the Markov model, because although we have a mobility model Π for intra-trajectory movement, we need a *start state* and an *end state* for specifying the two endpoints of our walk. Without the trip distribution we can assume a uniform distribution of start and end states, however, as Figure 4 shows, the trip distributions of real datasets are often heavily skewed. Thus, assuming a uniform distribution in place of the actual distribution is far from ideal, and will result in bogus synthetic trips that jeopardize utility and authenticity.

Assume that we discretize $\Omega(D_{real})$ using grid $\mathbb{A}$. We denote a trip using its start cell C_{start} and destination cell C_{end}, as: $C_{start} \rightsquigarrow C_{end}$. Let $h(D_{real}, C_{start} \rightsquigarrow C_{end})$ be a function that computes the number of trips $C_{start} \rightsquigarrow C_{end}$ in database D_{real}, and $\hat{h}$ denote its private noisy version. For brevity we drop D_{real} from the notation and simply write $h(C_{start} \rightsquigarrow C_{end})$. Then, treating X as a random variable over the domain of the trip distribution $\mathbb{A} \times \mathbb{A}$, the entries in the trip distribution denoted $\mathcal{R}$ are calculated as:

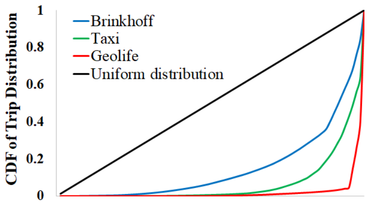

Figure 4: Cumulative Distribution Function (CDF) of the trip distribution $\mathcal{R}$ of 3 datasets, compared with the CDF of uniform distribution. For consistency we enforced a 3x3 uniform grid on all datasets and ordered the cell pairs on x-axis in non-decreasing trip frequency.

$$Pr\Big(X = (C_{start}, C_{end})\Big) = \begin{cases} \frac{\hat{h}(C_{start} \rightsquigarrow C_{end})}{\sum_{C_i}\sum_{C_j}\hat{h}(C_i \rightsquigarrow C_j)} & \text{for } C_{start}, C_{end} \in \mathbb{A} \\ 0 & \text{otherwise} \end{cases}$$

Note that our definition ensures $\mathcal{R}$ is a probability mass function (pmf) since its entries sum to 1.

In the case of a two-layer grid where one GPS location is indexed by both a top-level and bottom-level cell, we can use constrained inference for improved accuracy and consistency [15, 25, 43]. In particular, we employ the following linear Ordinary Least Squares (OLS) approach. We denote by C_i a cell in the top-level of the grid, and by $C_{i,j}$ a cell in the bottom-level of the grid where $1 \leq j \leq M_i^2$ (see the notation in Figure 3). We use budget $\theta\varepsilon_3$ when obtaining top-level trip counts $\hat{h}(C_i \rightsquigarrow C_j)$, and budget $(1-\theta)\varepsilon_3$ when obtaining bottom-level counts $\hat{h}(C_{i,k} \rightsquigarrow C_{j,l})$. Note that if we had no privacy or perturbation requirement, and we simply used the noise-free counts h instead of $\hat{h}$, the following would hold: $h(C_i \rightsquigarrow C_j) = \sum_k \sum_l h(C_{i,k} \rightsquigarrow C_{j,l})$. However, this may not hold after each h is perturbed with random noise. To re-establish consistency and minimize noise impact, OLS asserts that given the noisy values of $\hat{h}$, we can obtain optimized trip counts, denoted $\hat{h}'(C_i \rightsquigarrow C_j)$, as:

$$\begin{aligned} \hat{h}'(C_i \rightsquigarrow C_j) = &\frac{\theta^2 M_i^2 M_j^2}{(1-\theta)^2 + \theta^2 M_i^2 M_j^2} \cdot \hat{h}(C_i \rightsquigarrow C_j) \\ &+ \frac{(1-\theta)^2}{(1-\theta)^2 + \theta^2 M_i^2 M_j^2} \cdot \sum_k \sum_l \hat{h}(C_{i,k} \rightsquigarrow C_{j,l}) \end{aligned}$$

When optimizing the counts for the bottom-level, differences in optimized top-level counts calculated above are distributed equally among the bottom-level cells:

$$\begin{aligned} \hat{h}'(C_{i,k} \rightsquigarrow C_{j,l}) = &\hat{h}(C_{i,k} \rightsquigarrow C_{j,l}) \\ &+ \frac{\hat{h}'(C_i \rightsquigarrow C_j) - \sum_s \sum_t \hat{h}(C_{i,s} \rightsquigarrow C_{j,t})}{M_i^2 M_j^2} \end{aligned}$$

Finally, optimized trip counts $\hat{h}'$ are used in place of $\hat{h}$ in the definition of $\mathcal{R}$. For the derivation of OLS and a worked example, we refer the reader to Appendix B.

4.4 Route Length Distribution

The final component in AdaTrace's private synopsis consists of fitted distributions of route lengths. We fit a different theoretical

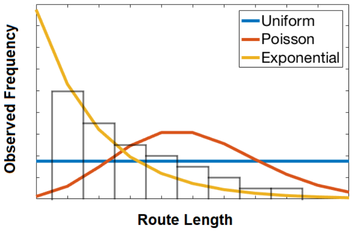

Figure 5: Comparing frequencies of observed route lengths (histogram) with candidate distributions.

distribution for each different trip $C_{start} \rightsquigarrow C_{end}$. The distribution is learned based on observed route lengths in D_{real} that make this trip. Our rationale in enforcing trip-specific length distributions is that trajectories travelling between certain regions may take more indirect routes than others, for reasons such as unavailability of roads, traffic avoidance, and so on. Hence, trip-specific length distributions will be more accurate than global distributions (learned using whole D_{real}) used in previous works [36, 55].

We treat observed route lengths as a histogram. We consider multiple well-known distributions with different shapes, such as uniform, exponential and Poisson distributions, as candidates to capture the shape of our histogram. An example is shown in Figure 5. The candidate distributions have one characteristic in common: Their parameters are directly related to aggregate summary statistics that can be privately and accurately obtained from D_{real}. For example, the Poisson distribution has a single parameter that is equal to the mean length, whereas the λ parameter of the exponential distribution is related to the median length *med* as: $\lambda = {}^{ln2}/_{med}$. We then use the Laplace and Exponential mechanisms to privately fetch statistics such as means and medians. A private mean can be fetched by decomposing it into a noisy sum divided by a noisy count, where the Laplace mechanism is used to inject noise. A private median can be fetched using Cormode et al.'s adaptation of the Exponential mechanism with scoring function $q := -|rank(x) - rank(med)|$ [15]. The mechanism returns noisy median x instead of the actual median, and the intuition behind this particular q is that if x is close to the actual median, then its rank will be similar to the median's rank. Thus, the score of each candidate is negatively impacted by how much its rank deviates from $rank(med)$.

After building multiple candidate distributions as above, we select one distribution as the best fit, store it in AdaTrace's memory, and drop the remaining. We use a *goodness of fit* test to determine which distribution is the best fit. While there are several tests to measure goodness of fit, we use the value of the χ^2 test statistic since its differentially private implementation is well known [22]. For example, in Figure 5 the goodness of fit test would select the exponential distribution as the best fit, since its shape is closest to the shape of the histogram.

4.5 Trajectory Synthesis Algorithm

AdaTrace's synthesis algorithm combines the four features in its private synopsis: the density-aware grid $\mathbb{A}$, the mobility model $\Pi(D_{real})$, the trip distribution $\mathcal{R}$ and the collection $\mathcal{L}$ of length distributions for each trip. It outputs synthetic trajectories based on the skeleton presented in Algorithm 1.

Algorithm 1: Trajectory synthesis algorithm

```
Input  : Grid $\mathbb{A}$, trip distribution $\mathcal{R}$, mobility model $\Pi$, length
         distributions $\mathcal{L}$
Output : A candidate synthetic trajectory $T_{syn}$
Pick a random sample $(C_{start}, C_{end})$ from pmf of $\mathcal{R}$
Retrieve from $\mathcal{L}$ the fitted probability distribution $PD$ for trip
  $C_{start} \rightsquigarrow C_{end}$
Pick a random sample $\ell$ from $PD$
Initialize $T_{syn}$ with $T_{syn}[1] = C_{start}$ and $T_{syn}[\ell] = C_{end}$
for $i = 2$ to $\ell - 1$ do
    for $C_{cand} \in \mathbb{A}$ do
        Retrieve from $\Pi$:
          $w_1 = Pr\Big(T[\text{i}] = C_{cand} \mid T[1] \ldots T[\text{i-1}]\Big)$ and
          $w_2 = Pr\Big(T[\ell] = C_{end} \mid T[1] \ldots T[\text{i-1}]C_{cand}\Big)$
        Set the weight of $C_{cand}$ equal to $w_1 \cdot w_2$
    end
    Sample $C_{chosen}$ from $\mathbb{A}$ with probability proportional to
      its weight calculated above
    Set $T_{syn}[\text{i}] = C_{chosen}$
end
return $T^{syn}$
```

The algorithm can be studied in four steps. First, it determines the start and end points of the synthetic trajectory T_{syn} by sampling from the trip distribution. Second, it determines the length of T_{syn} by sampling from the appropriate route length distribution in $\mathcal{L}$. Third, it initializes T_{syn} with first location as the starting cell of the trip, and last location as the destination cell of the trip. Fourth, given the two endpoints of T_{syn}, intermediate locations are found using a random walk on the mobility model Π. When determining the i'th position of T_{syn} considering the cells of grid $\mathbb{A}$ as candidates, each candidate is given a *weight* that consists of two sub-weights denoted w_1 and w_2. w_1 performs a look-back and finds the probability that the next location is C_{cand} given the previous locations, as in the straightforward application of the Markov assumption. This is essentially a one-step transition probability. On the other hand, w_2 performs a look-ahead and finds the probability that the final location is C_{end} given the previous locations and assuming the current location is set to C_{cand}. This is an $(\ell - i)$-step transition probability, which is computed using a combination of 1-step transition probabilities. To improve efficiency, we pre-compute multiple-step transition probabilities after learning Π so that the same computation is not repeated for different T_{syn}.

The above generates a trajectory for a *single* trip between well-defined start and end locations. This is sufficient when each user's GPS record in D_{real} corresponds to a short-term trip, such as an Uber or taxi ride. However, if D_{real} is collected over long periods of time (e.g., several days), then a user's record may contain multiple trips. In this case, the synthesis algorithm can be run multiple times per user, such that in each iteration, the starting location of the next trajectory is equal to the last known location of the previous trajectory. Then, these trajectories can be stitched together to form the final GPS record of the user with a desired number of trips.

5 PRIVACY ANALYSIS

AdaTrace has differential privacy and attack resilience as its two privacy goals. In this section we give concrete definitions and privacy parameters with which AdaTrace fulfills these goals. Our definitions are parameterized to enable the data owner to explicitly control the leakage potential of synthetic trajectories by specifying the desired privacy parameters.

5.1 Differential Privacy Preservation

AdaTrace satisfies ε-differential privacy as a whole. Recall that we treat ε as the total privacy budget and distribute it to four sub-budgets (one for each feature in the synopsis) such that $\sum_{i=1}^{4} \varepsilon_i = \varepsilon$. Learning and perturbing a feature consumes the ε_i allocated to it, thus depleting the total ε after the perturbation phase is complete, according to the sequential composition property. Then, any additional data-independent perturbation performed to satisfy attack resilience counts as post-processing. Note that sequential composition holds even when subsequent computations incorporate the outcomes of preceding computations [35], hence the fact that AdaTrace uses the perturbed grid in subsequent steps (Markov chain, trip distribution) does not violate ε-differential privacy. Also note that AdaTrace's synthesis algorithm performs sampling and calculation on perturbed features without accessing the actual database D_{real}. As a result, AdaTrace remains ε-differentially private.

Distribution of ε into ε_i can be done either by AdaTrace automatically or according to the specifications of the data owner. The current implementation of AdaTrace comes with a default budget distribution, which we empirically determined to yield high average utility: $\varepsilon_1 = \varepsilon/9$ for the grid, $\varepsilon_2 = 4\varepsilon/9$ for the Markov mobility model, $\varepsilon_3 = 3\varepsilon/9$ for the trip distribution and $\varepsilon_4 = \varepsilon/9$ for the length distribution. We can observe from this that not all features require high budgets to be accurate, and allocating an unnecessarily high budget to such features will negatively impact utility since it would steal from those features that do require high budgets to remain accurate. For example, the grid and length distribution components need lower budgets in comparison to others. One way to further optimize budget distribution is to leverage regression-based learning on D_{real} and ε, a plan for future work.

Although the above strategy is beneficial from a utility maximization perspective, we should also discuss its privacy implications. Since each individual feature will satisfy ε_i-differential privacy, a higher ε_i will cause the feature to be accurately preserved, whereas a lower ε_i will cause it to be more perturbed. A variable budget distribution allows the data owner (e.g., service provider) to distribute ε in a way that reflects which artifact he perceives is more sensitive and should thus be more perturbed to protect its privacy. For example, if the data owner feels that spatial densities are more sensitive, then a lower ε_1 can be assigned to grid construction. If trip distributions are sensitive, a lower ε_3 can be assigned, causing a more perturbed trip distribution. This provides flexibility with respect to different perceptions regarding what needs to be protected.

5.2 Enforcement of Attack Resilience

We now discuss how AdaTrace ensures attack resilience against the 3 privacy threats listed in Section 3.3.

Bayesian Inference. Recall that we denote by Z a sensitive zone such as a hospital, health clinic or religious place, by $\mathcal{B}(Z)$ the prior belief of an adversary regarding the users who visit Z, and by $\mathcal{B}(Z|D_{syn})$ the posterior belief having observed D_{syn}. The threat stems from $\mathcal{B}(Z|D_{syn})$ being significantly different than $\mathcal{B}(Z)$. To combat the threat, AdaTrace enforces the following defense.

Defense 1 (ϑ). *Denoting by EMD the Earth Mover's Distance, we say that D_{syn} is attack-resilient if the following holds, and susceptible otherwise.*

$$EMD\big(\mathcal{B}(Z), \mathcal{B}(Z|D_{syn})\big) \leq \vartheta$$

Intuitively, the privacy guarantee is that upon observing D_{syn}, an adversary's belief regarding users visiting Z does not change significantly, where the level of significance is controlled by parameter ϑ. This prohibits both positive and negative disclosures, i.e., the adversary is prohibited not only from learning that significantly higher than 10% of Z's visitors live in a certain neighborhood, but also that significantly lower than 10% of Z's visitors live in a certain neighborhood. The latter is useful when, for instance, Z is a university and the adversary may infer the lack of education.

AdaTrace follows an iterative perturbation strategy to implement Defense 1. Let D_Z denote the subset of trajectories that visit zone Z. We perturb the features concerning D_Z in the private synopsis, e.g., $\Pi(D_Z)$, $\mathcal{R}(D_Z)$, $\mathcal{L}(D_Z)$. In the most relaxed setting of $\vartheta = +\infty$ no perturbation is necessary; whereas in the strictest setting $\vartheta = 0$, the features must be maximally perturbed so that they exactly equal population averages. For any ϑ in between, we perturb the features iteratively, converging to population averages in each iteration, until Defense 1 is satisfied. After the defense is satisfied, we plug the perturbed features back to the private synopsis.

Partial Sniffing. Armed with user u's subtrajectory from the sniff region, the adversary aims to link u with a synthetic trajectory $T_s \in D_{syn}$. Since the user's actual trajectory T_u is not in D_{syn} the linkage is not genuine, however the threat stems from two cases: (i) T_s sufficiently resembles T_u, thus the adversary can still make correct inferences using T_s. (ii) T_s contains visits to a sensitive zone Z – although u may or may not have visited Z in reality, such an inference can have discriminating or harmful impact on u, and therefore should be prohibited.

Defense 2 (φ, ϱ). *Let SR denote the sniff region, $T_{sniffed}$ denote the sniffed subtrajectory, Z denote a sensitive zone, and visit(T, Z) denote the number of times trajectory T visits zone Z.*

1. *Find the matching synthetic trajectory:*
$$T_s = \underset{T \in D_{syn}}{\text{argmin}} \ DTW((T \cap SR), T_{sniffed})$$
2. *If $|intersection(T_s, T_u)| > \varphi$, mark as susceptible.*
3. *If $visit(T_s, Z) > \varrho$, mark as susceptible.*
4. *If no susceptibility is found in the above steps, mark as attack-resilient.*

AdaTrace enforces the above defense to thwart the partial sniffing threat. Its first step is to identify T_s using Dynamic Time Warping (DTW) as the geographical distance metric, which is a prominent method for measuring trajectory distance [30, 50, 52]. Then, in step 2, we prohibit correct location disclosures. Since exactly matching GPS coordinates and trajectories are rare, we allow for inexact intersections within a small error margin. φ is a threshold controlling the maximum amount of intersection allowed between T_s and T_u. It can be defined using absolute length (e.g., 100 meters, 1 kilometer) or using percentage of relative trajectory length (e.g., 10% of T_s). In step 3, we prohibit sensitive location disclosures. To prohibit *any* sensitive disclosure, we set $\varrho = 0$, which is suitable when trajectories are short, e.g., corresponding to taxi trips or home-work commutes. If trajectories are collected over long periods of time such as one month, it is advisable to set higher ϱ.

Outlier Leakage. In the outlier leakage attack, the threat stems from an adversary's ability to make sensitive inferences from atypical, outlier trajectories. To combat this threat, we must first identify and detect outliers. We adapt the most popular distance-based outlier definition [40, 44]: Outliers are those trajectories that have highest distance to their respective k'th nearest neighbor. It remains to define an appropriate distance function d so that distances between trajectories can be measured, and the nearest neighbors of each trajectory (according to d) can be identified. AdaTrace considers three distance functions, each of which corresponds to a different *type* of outlier:

(1) Trip outlier: Trajectories with non-traditional trip start and end locations. In this case, d is set as the geographical distance between trajectories' trip start and end locations.

(2) Length/duration outlier: Trajectories with unusually long or short lengths. In this case, d is set to measure the difference between two trajectories' route lengths.

(3) Mobility outlier: Trajectories with unusual mobility patterns, in which case d is set to measure the Jensen-Shannon Divergence (JSD) between the mobility models of two trajectories: $d(T_1, T_2) = \text{JSD}(\Pi(T_1), \Pi(T_2))$.

The distance functions above allow us to compute distances between pairs of trajectories and detect whether a trajectory is a certain type of outlier based on distance to its k'th nearest neighbor. After detecting outliers, AdaTrace enforces Defense 3.

Defense 3 (β, κ). *Let $T_{out} \in D_{syn}$ be a distance-based outlier, and d be the distance function used in detecting T_{out}.*

1. *Find the trajectory in D_{real} that is most similar to T_{out} with respect to d: $T_{akin} = \text{argmin}_{T \in D_{real}}\ d(T_{out}, T)$*
2. *Let y be the number of trajectories $T_{sim} \in D_{real}$ that satisfy: $d(T_{out}, T_{sim}) - d(T_{out}, T_{akin}) \leq \beta$*
3. *If $y \geq \kappa$, mark as attack-resilient, otherwise susceptible.*

The intuition behind this defense is as follows. We first identify T_{akin}, the real trajectory most similar to T_{out}. We then check how many real trajectories exist which have distance similar to T_{akin}'s distance to T_{out}, and denote this quantity by y. This effectively searches for a crowd of y trajectories with maximum distance β to the candidate outlier. Note that we always have $y \geq 1$ due to $T_{sim} = T_{akin}$ in step 2. Finally, if $y \geq \kappa$, where κ is the privacy parameter enforcing a minimum crowd size, we conclude that T_{out} plausibly blends in a crowd of actual trajectories. Hence, the trajectory is similar to a *group* of actual users' trajectories, it cannot be conclusively linked to a particular user, and outlier leakage is not a concern. However, if $y < \kappa$, i.e., T_{out} does not blend into a group of trajectories, then T_{out} must be discarded and replaced with a non-outlier trajectory.

Table 1: Datasets used in our experiments (μ:mean, σ:stdev)

	$\lvert D_{real} \rvert$	$\Omega(D_{real})$	$\lvert T \rvert$ ($\mu \pm \sigma$)	GPS samp. rate
Geolife	14,650	Beijing	931.4 ± 1602.5	~ 2.5 sec
Taxi	30,000	Porto	43.1 ± 33.5	~ 15 sec
Brinkhoff	50,000	Oldenburg	64.6 ± 35.9	~ 15.6 sec

6 EXPERIMENTAL EVALUATION

6.1 Experiment Setup

Implementation Details and Competitors. We implemented AdaTrace in Java. We compare it with 3 state of the art location trace generators: ngram, DPT and SGLT. We obtained their implementations from the respective authors [7, 11, 26]. In our comparison we use the parameters recommended by the authors when applicable, otherwise we experiment with different parameter values and report only the best results. Since all generators are probabilistic, each experiment is repeated 5 times and results are averaged.

We note the configuration used for ngram and SGLT: For ngram, although it is originally designed for sequential data generation, it has been used in [26] for comparison against DPT and in our experiments we adopt their setting, i.e., we discretize trajectories at different granularities, run ngram on discretized sequences and convert output synthetic sequences back to trajectories. We repeat this process at different granularities of discretization, and use the results of the granularity that yields highest accuracy. For SGLT, it is required that the input set of trajectories must be of same duration. In real life and in our datasets, trajectories vary in duration and sampling rate. To be able to run experiments with SGLT, we repeated the last known locations of shorter trajectories several times so that all trajectories would have equal duration.

Datasets. We used three datasets in our experiments: Geolife, Taxi and Brinkhoff. Geolife and Taxi are real datasets, whereas Brinkhoff is simulated. Information regarding the datasets is given below and in Table 1. Geolife was collected and released by Microsoft as part of the Geolife project [56]. It contains GPS traces of 182 users over 5 years. Majority of the data is from Beijing, China, with few outliers in other cities in China, Europe, etc. We pre-processed the data to remove these outliers and obtain a more even data distribution and well-defined $\Omega(D)$. The resulting dataset contained 14,650 trajectories. Taxi contains GPS traces of taxis operating in the city of Porto, Portugal. We extracted 30,000 trips from the denser areas in the dataset made available as part of the Taxi Service Prediction Challenge at ECML-PKDD 2015 [38]. Brinkhoff contains trajectories of vehicles simulated using Brinkhoff's network generator for moving objects [9]. The map of Oldenburg, Germany was used to simulate movements of 50,000 vehicles. Locations were sampled at equal time intervals.

6.2 Evaluation Metrics

We generated synthetic databases D_{syn} with cardinality $|D_{syn}| = |D_{real}|$, and used the following metrics to quantify the difference (or resemblance) between D_{real} and D_{syn}. When keeping the privacy level constant, lower difference (resp. higher resemblance) is desired for improved utility. According to Section 3.4, the three trajectory utility categories we aimed to preserve in AdaTrace were spatial densities and localities, spatio-temporal travel metrics, and frequent travel patterns. The evaluation metrics we present in this section fall under these categories, as described below.

Spatial density and locality metrics. A large number of geodata analysis tasks, including PoI and popular location extraction, hotspot discovery, heatmaps and recommendation engines rely on spatial densities. We employ the following two metrics to measure the quality of spatial density and locality resemblance.

Our first metric is *query error*, which is a popular measure for evaluating data synthesis algorithms ranging from tabular data to location and graph data [11, 12, 33]. We consider spatial counting queries of the form: "Retrieve the number of trajectories passing through a certain region R". Let Q denote a query of this form, and $Q(D)$ denote its answer when issued on database D. The relative error (RE) of Q is defined as:

$$RE = \frac{|Q(D_{real}) - Q(D_{syn})|}{max\{Q(D_{real}), b\}}$$

where b is a sanity bound to mitigate the effect of extremely selective queries. We set $b = 1\% \times |D|$. We generate 500 random queries by choosing their regions R uniformly at random and compute average relative error (AvRE) by averaging the RE of all queries.

Our second metric is for measuring the discrepancies in locations' popularity ranking. Let $L_1, ..., L_n$ be a list of locations and let $pop(D, L_i)$ measure the popularity of L_i, i.e., number of times L_i is visited by trajectories in D. We compute $pop(D_{real}, L_i)$ and $pop(D_{syn}, L_i)$ for all L_i. Then, we say that a pair of locations (L_i, L_j) are *concordant* if either of the following hold:

$$(pop(D_{real}, L_i) > pop(D_{real}, L_j)) \wedge (pop(D_{syn}, L_i) > pop(D_{syn}, L_j))$$
$$(pop(D_{real}, L_i) < pop(D_{real}, L_j)) \wedge (pop(D_{syn}, L_i) < pop(D_{syn}, L_j))$$

That is, their popularity ranks (in sorted order) agree. They are said to be *discordant* if their ranks disagree. The Kendall-tau coefficient can then be applied as:

$$KT = \frac{(\text{\# of concordant pairs}) - (\text{\# of discordant pairs})}{n(n-1)/2}$$

In our experiment, we determine the locations $L_1, ..., L_n$ using a fine-grained grid and use $n = 400$ locations.

Frequent travel pattern metrics. Mining popular travel patterns has also received significant attention from the geodata analysis community, for understanding traffic flows and road network performance, and providing better navigation services. The following two metrics measure resemblance with respect to frequent patterns (FPs). For both metrics, we project trajectories on a uniform grid $\mathcal{U}$, thus obtaining the sequence of cells they pass through. We define a pattern P as an ordered sequence of cells, e.g., $P : C_2 \to C_4 \to C_3$. We define the support of a pattern, $supp(D, P)$, as the number of occurrences of P in database D. We mine the top-k patterns in D, i.e., the k patterns with highest support, denoted by $\mathcal{F}^k_{\mathcal{U}}(D)$.

Our first metric measures difference in patterns' support. We calculate the relative difference between $supp(D_{real}, P)$ and $supp(D_{syn}, P)$ for each frequent pattern $P \in \mathcal{F}$. Formally, the FP average relative error is:

$$FP\ AvRE = \frac{\sum_{P \in \mathcal{F}^k_{\mathcal{U}}(D_{real})} \frac{|supp(D_{real}, P) - supp(D_{syn}, P)|}{supp(D_{real}, P)}}{k}$$

Our second metric measures the set similarity between the top-k patterns in D_{real} and the top-k patterns in D_{syn}. Following the F1-measure, i.e., harmonic mean of precision and recall, we define the FP similarity metric as:

$$FP\ Similarity = \text{F1}(\mathcal{F}^k_{\mathcal{U}}(D_{real}), \mathcal{F}^k_{\mathcal{U}}(D_{syn}))$$

FP similarity score is between 0 and 1, where higher score implies better set preservation, and is therefore more desired. For the AvRE metric, lower error values are more desired. In our experiments we use $k = 100$, as higher k returned patterns with insignificant support. Also, we assume a 6x6 uniform grid for $\mathcal{U}$ and only consider patterns that are at least 3 cells long.

Spatio-temporal travel metrics. Since many trajectory databases consist of taxi/Uber rides or daily commutes, analysis of spatio-temporal features of these trips becomes critical. We employ 3 evaluation metrics for trip and travel analysis.

Our first metric, called *trip error*, measures how well the correlations between trips' start and end regions are preserved. We make use of the empirical trip distribution $\mathcal{R}$ defined in Section 4.3. Given the grid $\mathcal{U}$, we compute the trip distributions of the real and synthetic databases, denoted $\mathcal{R}(D_{real})$ and $\mathcal{R}(D_{syn})$ respectively. The trip error is defined as: JSD($\mathcal{R}(D_{real}), \mathcal{R}(D_{syn})$) where JSD is the Jensen-Shannon divergence.

Our second metric, called *length error*, measures the error in trip lengths (i.e., distances travelled in each trip). We calculate the total distance travelled in a trip by adding up the distance between each consecutive GPS reading. Upon learning the maximum length from D_{real}, we quantize trip lengths into 20 equi-width buckets: $\{[0, x), [x, 2x), ..., [19x, 20x]\}$, where $20x$ is the longest length present in D_{real}. For each bucket we determine how many trips' length fall into that bucket, thereby obtaining a histogram of lengths. Let $\mathcal{N}(D_{real})$ and $\mathcal{N}(D_{syn})$ denote this empirical, bucketized histogram of real and synthetic databases respectively. The length error is calculated as: JSD($\mathcal{N}(D_{real}), \mathcal{N}(D_{syn})$).

Our final metric, called *diameter error*, is adopted from [26]. The diameter of a trajectory T is defined as the maximum distance between any pair of its GPS readings (not necessarily consecutive). Similar to length error, we learn the maximum diameter from D_{real}, perform equi-width bucketing and histogram extraction. Let $\mathcal{E}(D_{real})$ and $\mathcal{E}(D_{syn})$ denote the diameter distributions of the real and synthetic database respectively. The diameter error is calculated as: JSD($\mathcal{E}(D_{real}), \mathcal{E}(D_{syn})$).

6.3 Comparison with Existing Generators

In this section we compare AdaTrace with ngram, DPT and SGLT. Since all generators except SGLT have ε (differential privacy budget) as a parameter, we perform our comparison by varying ε. Results are summarized in Table 2. We make two general observations. First, AdaTrace provides superior utility when subjected to the same level of privacy: Out of 63 total comparison settings, AdaTrace outperforms its competitors in 53 cases. In certain settings, AdaTrace's utility improvement is significant (2-3 fold or more). In addition to a utility advantage, AdaTrace also has a privacy advantage over its competitors due to its attack resilience property. Second, we observe that errors decrease and utility increases when higher ε is used. As such, we can conclude that data quality is responsive to changes in the privacy parameters, making AdaTrace flexible.

Table 2: Comparing AdaTrace with its competitors. Best result in each category is shown in bold. For FP F1 Similarity and location Kendall-tau, higher values are better. For remaining metrics, lower values are better.

		Geolife				Taxi				Brinkhoff			
		ngram	DPT	SGLT	AdaTrace	ngram	DPT	SGLT	AdaTrace	ngram	DPT	SGLT	AdaTrace
Query AvRE	ε=0.5	0.391	0.158	**0.123**	0.168	0.232	0.381	0.158	**0.091**	0.201	0.386	0.216	**0.151**
	ε=1.0	0.383	0.161	**0.123**	0.162	0.233	0.380	0.158	**0.089**	0.158	0.342	0.216	**0.142**
	ε=2.0	0.355	0.154	**0.123**	0.155	0.222	0.369	0.158	**0.088**	0.142	0.291	0.216	**0.138**
FP AvRE	ε=0.5	0.60	0.58	0.75	**0.47**	0.43	0.65	0.53	**0.41**	0.83	0.53	0.52	**0.39**
	ε=1.0	0.61	0.57	0.75	**0.41**	0.42	0.69	0.53	**0.37**	0.64	0.48	0.52	**0.38**
	ε=2.0	0.59	0.56	0.75	**0.41**	0.43	0.68	0.53	**0.36**	0.54	**0.36**	0.52	0.38
Trip Error	ε=0.5	0.418	0.339	0.143	**0.048**	0.225	0.492	0.279	**0.042**	0.156	0.170	0.298	**0.052**
	ε=1.0	0.412	0.341	0.143	**0.025**	0.220	0.561	0.279	**0.033**	0.150	0.120	0.298	**0.045**
	ε=2.0	0.317	0.296	0.143	**0.013**	0.236	0.514	0.279	**0.027**	0.139	0.106	0.298	**0.043**
Length Error	ε=0.5	0.020	0.131	0.173	**0.011**	0.085	0.020	0.106	**0.017**	0.094	0.057	0.126	**0.042**
	ε=1.0	0.021	0.124	0.173	**0.010**	0.060	0.017	0.106	**0.016**	0.057	0.051	0.126	**0.041**
	ε=2.0	0.019	0.125	0.173	**0.008**	0.025	0.019	0.106	**0.014**	**0.032**	0.038	0.126	0.039
Diameter Error	ε=0.5	0.125	0.255	0.170	**0.085**	0.239	0.372	0.129	**0.026**	0.151	0.143	0.172	**0.022**
	ε=1.0	0.124	0.251	0.170	**0.067**	0.229	0.374	0.129	**0.025**	0.103	0.084	0.172	**0.023**
	ε=2.0	0.125	0.249	0.170	**0.065**	0.234	0.373	0.129	**0.024**	0.092	0.061	0.172	**0.022**
FP F1 Similarity	ε=0.5	0.38	0.46	0.38	**0.57**	0.42	0.27	0.49	**0.66**	0.41	0.56	0.47	**0.61**
	ε=1.0	0.42	0.55	0.38	**0.61**	0.54	0.31	0.49	**0.68**	0.39	**0.64**	0.47	0.62
	ε=2.0	0.41	**0.60**	0.38	**0.60**	0.61	0.32	0.49	**0.69**	0.42	**0.71**	0.47	0.62
Loc. Kendall-tau	ε=0.5	0.53	0.39	0.62	**0.71**	0.68	0.45	0.69	**0.82**	0.71	0.68	0.61	**0.73**
	ε=1.0	0.55	0.44	0.62	**0.68**	0.72	0.46	0.69	**0.83**	**0.76**	0.65	0.61	0.74
	ε=2.0	0.52	0.54	0.62	**0.69**	0.76	0.51	0.69	**0.83**	0.75	**0.86**	0.61	0.74

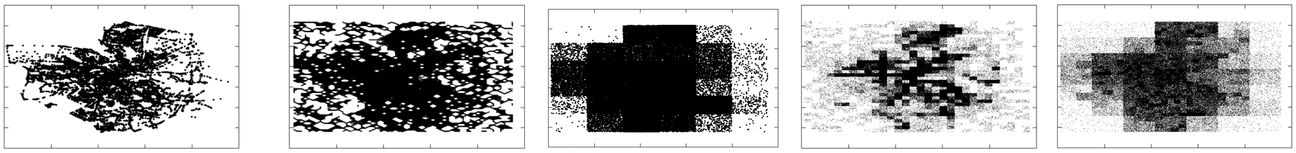
(a) Actual (b) DPT (c) SGLT (d) ngram (e) AdaTrace

Figure 6: Visual density distributions of synthetic databases generated by various generators, using Brinkhoff as their D_{real}.

Next, we perform an in-depth analysis by studying the results under each utility category one by one, starting with *spatial density and locality.* We expect that Query AvRE is heavily related to the spatial distribution of trajectories. Hence, upon observing the numerical results in Table 2 we performed the visual comparison illustrated in Figure 6, where we compare the density distributions of synthetic databases produced by various trajectory generators using Brinkhoff as input. Figure 6 validates the finding in Table 2 that AdaTrace does better in terms of spatial density. DPT and SGLT's errors stem from low-density or medium-density regions becoming exceedingly dense, and ngram's errors stem from medium-density regions becoming exceedingly sparse. AdaTrace's density-adaptive grid plays an important role in density preservation, which we show in more detail in Appendix C. Another interesting note is that although Query AvRE and location Kendall-tau metrics have a positive correlation, they do not necessarily agree in all settings. For example, observe that SGLT's AvRE is lowest on Geolife, however, AdaTrace performs better than SGLT in terms of Kendall-tau. This demonstrates that there can be benefit in using multiple evaluation metrics for measuring one type of trajectory utility.

In terms of *frequent travel patterns*, there is correlation between FP AvRE and F1 similarity metrics, but also some discrepancies. AvRE indicates that there is 30-40% error in FP support when using AdaTrace which might seem high, but we observed that this is caused mostly by FPs with highest support not having as high support in synthetic data. That is, even though an actually frequent FP is correctly identified as frequent in synthetic data, since its frequency is not as high, considerable support error is incurred. This explains the competitive F1 scores despite high error in AvRE.

In terms of *spatio-temporal travel metrics*, we first note that AdaTrace is significantly superior in terms of trip error. Second, we notice that although AdaTrace's length errors are similar to its competitors, its diameter errors are much smaller. This is surprising, given that the two metrics are clearly related. To better understand this behavior, we individually studied some trajectories from each generator. We observed that DPT and ngram are biased towards generating trajectories with low displacement. For example, their trajectories contain loops and U-turns, which means that although they travel long distances, their total displacement is not large. In contrast, AdaTrace's synthesis algorithm prevents unrealistic loops and U-turns by guiding a trajectory's random walk with its final destination. That is, at each step of the random walk AdaTrace considers the final destination that the trajectory will eventually reach and how many steps are left to reach there. Hence, the resulting trajectory becomes goal-driven, which realistically captures

Table 3: Execution time measurements

	ngram	DPT	SGLT	AdaTrace
Geolife	7 mins	1.9 mins	11 hrs	0.7 mins
Taxi	12 mins	2.3 mins	2.4 days	1.7 mins
Brinkhoff	18 mins	3.5 mins	6.9 days	2.2 mins

the mobility of human beings since humans often travel with a destination in mind.

Finally, we comment on the efficiency of each approach. We performed our experiments on a high-end laptop with Intel i7 CPU, but since all generators are single-threaded, they utilized a single 2.8 GHz CPU. Also, 16 GB main memory was sufficient for the computation performed by all generators. In terms of execution time, we obtained the results given in Table 3. Overall, AdaTrace is very efficient – it is able to process and generate 50,000 trajectories in slightly longer than 2 minutes. ngram and DPT are also reasonably efficient since they finish in several minutes, but not as efficient as AdaTrace. On the other hand, SGLT is significantly slower, as it takes several days to produce the same number of trajectories AdaTrace can produce in 2 minutes. This greatly hinders its use on commodity hardware or mobile devices. Note that this observation is consistent with the authors' claims in their paper [7], as they also report up to 2 minutes per generated trajectory. This has not caused problems in their experiments since their datasets consisted of < 100 users, but our datasets were several orders of magnitude larger (15-50k trajectories), surfacing an inefficiency problem.

6.4 Impact of Attack Resilience on Utility

In the previous section we conducted experiments with varying differential privacy parameter ε and observed its impact on utility. In this section we analyze the impact of attack resilience parameters on various forms of utility.

Defense 1: Bayesian Inference. Our defense asserts that the difference between priors and posteriors must be $\leq \vartheta$. When $\vartheta = 0$ the defense is strictest, and it is relaxed as ϑ increases. In Figure 7a, we initially set $\vartheta = 0$, increase it gradually, and report the percentage improvement (decrease) in Query AvRE and FP AvRE. We only report these metrics since the remaining metrics are non-linear, involving the likes of F1 or JSD calculation, hence percentages are not meaningful. The results show the expected trait that both spatial densities (implied by Query AvRE) and frequent patterns (implied by FP AvRE) are positively impacted when ϑ is relaxed. We observed that this is mostly caused by queries and FPs involving the sensitive zone Z, which is central to Defense 1. When the privacy setting is stricter, trajectories visiting Z are more random, hence FPs including Z will not be preserved.

An implicit parameter here is the choice (size and density) of Z. In the extreme case where $Z = \Omega(D_{real})$, Defense 1 dictates that the adversary should gain no knowledge that adds to his prior belief. In this case the trajectories in D_{syn} will be most random, and their features will converge to population averages rather than capturing the utility in D_{real}. To experimentally confirm this, in Appendix D we perform an experiment in which we vary the size of Z (see Figure 9a). Results indicate that there is indeed a positive correlation between error amounts and the size of Z.

Defense 2: Partial Sniffing. The defense is based on two parameters: φ that bounds the intersection between a sniffed actual trajectory and its counterpart synthetic trajectory, and ϱ that limits the possibility that the counterpart visits a sensitive zone. For both parameters, lower values imply stricter privacy. We set a fixed sniffing region that is visited by 4% of the actual trajectories, and assume all trajectories in this region have been sniffed by the adversary. We set $\varrho = 0$ which is the strictest privacy setting, and vary φ to observe its utility impact. In Figure 7b and 9b we illustrate the utility impact wrt 3 metrics. Although one could expect utility to decrease as privacy becomes stricter, this does not appear to be the case – we observe that aggregate utility stays constant despite changes in φ. Upon further analysis with more data points and remaining utility metrics, we confirm that the impact of φ on overall utility is negligible. This is because our utility metrics are aggregate metrics, taking whole D_{real} and D_{syn} into consideration rather than comparing individual trajectories one by one. This is also the case in many machine learning tasks and real-life uses of trajectory data. Thus, although an actual trajectory and its synthetic counterpart do not closely match, utility can be re-injected by implicitly adding the mismatched part into a different trajectory that is not sniffed.

Defense 3: Outlier Leakage. The two parameters in this defense are β and κ. κ dictates that a synthetic outlier must blend into a crowd of κ actual trajectories. β dictates the uniformity of the crowd: When $\beta = 0$ all trajectories in the crowd must appear exactly the same (in terms of the trajectory feature considered), whereas a large β allows the crowd to have higher non-uniformity. The privacy requirement becomes stricter when β decreases and κ increases.

We first analyze the behavior of Defense 3 wrt parameters β and κ in Figure 7c. Here, we set β normalized after observing the maximum possible distance in the database, e.g., if the max distance is 100, $\beta = 0.1$ implies that the allowed distance in forming the crowd is 10. When the privacy requirement is stricter, we indeed observe that there are more outlier trajectories failing the defense, with a notable increase when κ becomes ≥ 25. We attribute this to the curse of dimensionality: Crowd-blending notions of privacy, such as k-anonymity and our defense, rely on the ability to find a crowd comprised of similar records. While this is easier for low-dimensional data, trajectory data is inherently high-dimensional, and typically no two trajectories are alike. Hence, forming a similar crowd is difficult, and while higher κ values can be suitable for low-dimensional databases, the same κ may yield an excess number of outliers on a high-dimensional or trajectory database.

Having said that, we are interested in measuring the utility impact of arbitrary (β,κ) pairs, and wish that our system preserves utility even in the strictest cases. We therefore design the experiment in Figure 7d. Instead of trying to isolate β and κ, we take a holistic approach and base our utility analysis on the number of outliers failing the defense. Note that outliers failing the defense cannot be released, and must be perturbed or regenerated. We observe from the experiment results that this perturbation or regeneration process has little to no impact on aggregate trajectory utility. Despite different privacy settings, Query AvRE, Trip Error and Length Error do not seem to be impacted. Hence, we can conclude that we are able to protect against outlier leakage with no additional utility cost. In addition to the visual results in Figure 7d,

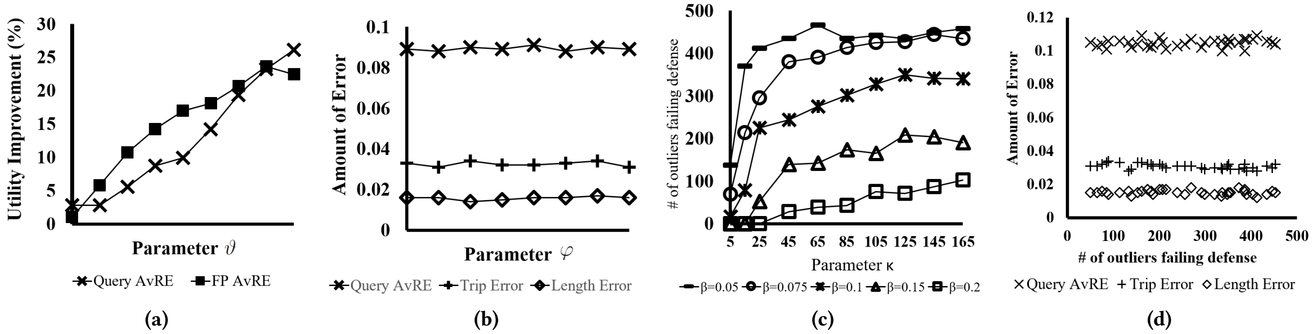

Figure 7: (a) Impact of ϑ on aggregate trajectory utility. (b) Impact of φ on aggregate trajectory utility. (c) Analyzing the outlier behavior with respect to parameters β, κ. (d) Utility impact of executing Defense 3. All experiments are performed on Taxi.

we computed the correlation between the privacy settings and the error amounts. Results indicate weak correlations at most (+0.2), but are often closer to 0 and sometimes even negative.

Summary of Findings. Under reasonable privacy parameters, Defenses 2 and 3 can be implemented with little or no observable impact on aggregate trajectory utility, because they are concerned with individual trajectories that often constitute a small portion of the whole database. The impact of small perturbations on individual trajectories are either negligible on aggregate utility, or they can be counterbalanced by injecting the lost utility in the remaining trajectories. On the other hand, Defense 1 is directly connected to the aggregate information that is obtained from D_{syn}, hence a stricter privacy parameter ϑ or an increased geographic size of sensitive zone Z negatively impacts aggregate trajectory utility.

7 CONCLUSION

We presented AdaTrace, a utility-aware location trace synthesizer with differential privacy guarantee and attack resilience. AdaTrace performs feature extraction, learning, and noise injection using a database of real location traces. It then generates synthetic traces while preserving differential privacy, enforcing resilience to inference attacks, and upholding statistical and spatial utility. Our experiments on real and simulated datasets show that AdaTrace is highly effective. Compared to ngram, DPT, and SGLT [7, 11, 26], AdaTrace provides up to 3-fold improvement in trace utility. At the same time, AdaTrace is computationally more efficient, and on average 1.5x faster than DPT, 8x faster than ngram, and 1000x faster than SGLT. One of our ongoing research directions is to investigate the feasibility of extending AdaTrace to handle incremental updates, to make it compatible with growing and streaming location trace datasets.

ACKNOWLEDGMENT

This research was partially sponsored by NSF under grants SaTC 1564097 and an RCN BD Fellowship, provided by the Research Coordination Network (RCN) on Big Data and Smart Cities and an IBM Faculty Award. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding agencies and companies mentioned above.

REFERENCES

[1] 2017. NYC Taxi & Limousine Commission – Trip Record Data. http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml. Accessed: 2018-05-03.

[2] 2018. Analyzing 1.1 Billion NYC Taxi and Uber Trips, with a Vengeance. http://toddwschneider.com/posts/analyzing-1-1-billion-nyc-taxi-and-uber-trips-with-a-vengeance/. Accessed: 2018-05-03.

[3] 2018. Uber Movement: Let's find smarter ways forward. movement.uber.com. Accessed: 2018-05-03.

[4] Gergely Acs and Claude Castelluccia. 2014. A case study: Privacy preserving release of spatio-temporal density in paris. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1679–1688.

[5] Miguel E Andrés, Nicolás E Bordenabe, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. 2013. Geo-indistinguishability: Differential privacy for location-based systems. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 901–914.

[6] Bhuvan Bamba, Ling Liu, Peter Pesti, and Ting Wang. 2008. Supporting anonymous location queries in mobile environments with privacygrid. In *Proceedings of the 17th international conference on World Wide Web*. ACM, 237–246.

[7] Vincent Bindschaedler and Reza Shokri. 2016. Synthesizing plausible privacy-preserving location traces. In *2016 IEEE Symposium on Security and Privacy (S&P)*. IEEE, 546–563.

[8] Nicolás E Bordenabe, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. 2014. Optimal geo-indistinguishable mechanisms for location privacy. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 251–262.

[9] Thomas Brinkhoff. 2002. A framework for generating network-based moving objects. *GeoInformatica* 6, 2 (2002), 153–180.

[10] Konstantinos Chatzikokolakis, Catuscia Palamidessi, and Marco Stronati. 2014. A predictive differentially-private mechanism for mobility traces. In *International Symposium on Privacy Enhancing Technologies*. Springer, 21–41.

[11] Rui Chen, Gergely Acs, and Claude Castelluccia. 2012. Differentially private sequential data publication via variable-length n-grams. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*. ACM, 638–649.

[12] Rui Chen, Benjamin CM Fung, S Yu Philip, and Bipin C Desai. 2014. Correlated network data publication via differential privacy. *The VLDB Journal* 23, 4 (2014), 653–676.

[13] Chris Clifton and Tamir Tassa. 2013. On syntactic anonymity and differential privacy. *Transactions on Data Privacy* 6, 2 (2013), 161–183.

[14] Graham Cormode. 2011. Personal privacy vs population privacy: learning to attack anonymization. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1253–1261.

[15] Graham Cormode, Cecilia Procopiuc, Divesh Srivastava, Entong Shen, and Ting Yu. 2012. Differentially private spatial decompositions. In *2012 IEEE 28th International Conference on Data Engineering (ICDE)*. IEEE, 20–31.

[16] Wei-Yen Day and Ninghui Li. 2015. Differentially private publishing of high-dimensional data using sensitivity control. In *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security (AsiaCCS)*. ACM, 451–462.

[17] Yves-Alexandre De Montjoye, César A Hidalgo, Michel Verleysen, and Vincent D Blondel. 2013. Unique in the crowd: The privacy bounds of human mobility. *Scientific Reports* 3 (2013), 1376.

[18] Cynthia Dwork. 2006. Differential Privacy. In *International Colloquium on Automata, Languages, and Programming*. Springer, 1–12.

[19] Cynthia Dwork. 2008. Differential privacy: A survey of results. In *International Conference on Theory and Applications of Models of Computation*. Springer, 1–19.

[20] Cynthia Dwork, Aaron Roth, et al. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science* 9, 3–4 (2014), 211–407.

[21] Matthew Fredrikson, Eric Lantz, Somesh Jha, Simon Lin, David Page, and Thomas Ristenpart. 2014. Privacy in Pharmacogenetics: An End-to-End Case Study of Personalized Warfarin Dosing. In *USENIX Security Symposium*. 17–32.

[22] Marco Gaboardi, Hyun-Woo Lim, Ryan M Rogers, and Salil P Vadhan. 2016. Differentially Private Chi-Squared Hypothesis Testing: Goodness of Fit and Independence Testing. In *ICML*. 2111–2120.

[23] Bugra Gedik and Ling Liu. 2005. Location privacy in mobile systems: A personalized anonymization model. In *25th IEEE International Conference on Distributed Computing Systems (ICDCS 2005)*. IEEE, 620–629.

[24] Arthur Gervais, Reza Shokri, Adish Singla, Srdjan Capkun, and Vincent Lenders. 2014. Quantifying web-search privacy. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 966–977.

[25] Michael Hay, Vibhor Rastogi, Gerome Miklau, and Dan Suciu. 2010. Boosting the accuracy of differentially private histograms through consistency. *Proceedings of the VLDB Endowment* 3, 1-2 (2010), 1021–1032.

[26] Xi He, Graham Cormode, Ashwin Machanavajjhala, Cecilia M Procopiuc, and Divesh Srivastava. 2015. DPT: Differentially private trajectory synthesis using hierarchical reference systems. *Proceedings of the VLDB Endowment* 8, 11 (2015), 1154–1165.

[27] Ari Juels and Ronald L Rivest. 2013. Honeywords: Making password-cracking detectable. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 145–160.

[28] Shiva P Kasiviswanathan and Adam Smith. 2014. On the'semantics' of differential privacy: A bayesian formulation. *Journal of Privacy and Confidentiality* 6, 1 (2014), 1.

[29] Ryo Kato, Mayu Iwata, Takahiro Hara, Akiyoshi Suzuki, Xing Xie, Yuki Arase, and Shojiro Nishio. 2012. A dummy-based anonymization method based on user trajectory with pauses. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*. ACM, 249–258.

[30] Eamonn Keogh and Chotirat Ann Ratanamahatana. 2005. Exact indexing of dynamic time warping. *Knowledge and Information Systems* 7, 3 (2005), 358–386.

[31] Daniel Kifer and Ashwin Machanavajjhala. 2011. No free lunch in data privacy. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*. ACM, 193–204.

[32] Daniel Kifer and Ashwin Machanavajjhala. 2014. Pufferfish: A framework for mathematical privacy definitions. *ACM Transactions on Database Systems (TODS)* 39, 1 (2014), 3.

[33] Haoran Li, Li Xiong, and Xiaoqian Jiang. 2014. Differentially private synthesization of multi-dimensional data using copula functions. In *International Conference on Extending Database Technology (EDBT)*, Vol. 2014. NIH Public Access, 475.

[34] Frank McSherry and Kunal Talwar. 2007. Mechanism design via differential privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science, (FOCS) 2007*. IEEE, 94–103.

[35] Frank D McSherry. 2009. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*. ACM, 19–30.

[36] Darakhshan J Mir, Sibren Isaacman, Ramón Cáceres, Margaret Martonosi, and Rebecca N Wright. 2013. Dp-where: Differentially private modeling of human mobility. In *2013 IEEE International Conference on Big Data*. IEEE, 580–588.

[37] Zarrin Montazeri, Amir Houmansadr, and Hossein Pishro-Nik. 2017. Achieving Perfect Location Privacy in Wireless Devices Using Anonymization. *IEEE Transactions on Information Forensics and Security* 12, 11 (2017), 2683–2698.

[38] Luis Moreira-Matias, Joao Gama, Michel Ferreira, Joao Mendes-Moreira, and Luis Damas. 2013. Predicting taxi–passenger demand using streaming data. *IEEE Transactions on Intelligent Transportation Systems* 14, 3 (2013), 1393–1402.

[39] Hoa Ngo and Jong Kim. 2015. Location privacy via differential private perturbation of cloaking area. In *2015 IEEE 28th Computer Security Foundations Symposium (CSF)*. IEEE, 63–74.

[40] Gustavo H Orair, Carlos HC Teixeira, Wagner Meira Jr, Ye Wang, and Srinivasan Parthasarathy. 2010. Distance-based outlier detection: consolidation and renewed bearing. *Proceedings of the VLDB Endowment* 3, 1-2 (2010), 1469–1480.

[41] Sai Teja Peddinti and Nitesh Saxena. 2010. On the privacy of web search based on query obfuscation: a case study of TrackMeNot. In *International Symposium on Privacy Enhancing Technologies*. Springer, 19–37.

[42] Apostolos Pyrgelis, Carmela Troncoso, and Emiliano De Cristofaro. 2018. Knock Knock, Who's There? Membership Inference on Aggregate Location Data. In *Network and Distributed System Security Symposium (NDSS) 2018*.

[43] Wahbeh Qardaji, Weining Yang, and Ninghui Li. 2013. Differentially private grids for geospatial data. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*. IEEE, 757–768.

[44] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. 2000. Efficient algorithms for mining outliers from large data sets. In *ACM SIGMOD Record*, Vol. 29. ACM, 427–438.

[45] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *IEEE Symposium on Security and Privacy (SP)*. IEEE, 3–18.

[46] Reza Shokri, George Theodorakopoulos, Jean-Yves Le Boudec, and Jean-Pierre Hubaux. 2011. Quantifying location privacy. In *2011 IEEE Symposium on Security and Privacy (S&P)*. IEEE, 247–262.

[47] Reza Shokri, George Theodorakopoulos, Carmela Troncoso, Jean-Pierre Hubaux, and Jean-Yves Le Boudec. 2012. Protecting location privacy: optimal strategy against localization attacks. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*. ACM, 617–627.

[48] Chaoming Song, Zehui Qu, Nicholas Blumm, and Albert-László Barabási. 2010. Limits of predictability in human mobility. *Science* 327, 5968 (2010), 1018–1021.

[49] Hien To, Kien Nguyen, and Cyrus Shahabi. 2016. Differentially private publication of location entropy. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 35.

[50] Kevin Toohey and Matt Duckham. 2015. Trajectory similarity measures. *SIGSPATIAL Special* 7, 1 (2015), 43–50.

[51] Yonghui Xiao and Li Xiong. 2015. Protecting locations with differential privacy under temporal correlations. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 1298–1309.

[52] Byoung-Kee Yi, HV Jagadish, and Christos Faloutsos. 1998. Efficient retrieval of similar time sequences under time warping. In *14th International Conference on Data Engineering (ICDE 1998)*. IEEE, 201–208.

[53] Tun-Hao You, Wen-Chih Peng, and Wang-Chien Lee. 2007. Protecting moving trajectories with dummies. In *2007 International Conference on Mobile Data Management*. IEEE, 278–282.

[54] Lei Yu, Ling Liu, and Calton Pu. 2017. Dynamic differential location privacy with personalized error bounds. In *Network and Distributed System Security Symposium (NDSS '17)*.

[55] Ling Zhang, Shuangling Luo, and Haoxiang Xia. 2016. An Investigation of Intra-Urban Mobility Pattern of Taxi Passengers. *arXiv preprint arXiv:1612.08378* (2016).

[56] Yu Zheng, Lizhu Zhang, Xing Xie, and Wei-Ying Ma. 2009. Mining interesting locations and travel sequences from GPS trajectories. In *Proceedings of the 18th International Conference on World Wide Web*. ACM, 791–800.

A SENSITIVITY OF GRID CONSTRUCTION

Recall that we access the trajectory database D while the density-adaptive grid is being constructed through normalized visit count queries denoted g, once per cell C_i where $i \in [1, N^2]$. This results in the query set $\boldsymbol{W} = \{g(D, C_1), g(D, C_2), ..., g(D, C_{N^2})\}$.

Also recall the definitions of sensitivity and neighboring databases from Section 3.2. A neighboring database implies that either $D = D' \cup \{T\}$ or $D' = D \cup \{T\}$, where T denotes a single user's location trajectory. Without loss of generality we assume $D' = D \cup \{\tau\}$. We need to show $\Delta\boldsymbol{W} = \max_{D,D'} ||\boldsymbol{W}(D') - \boldsymbol{W}(D)|| = 1$. The derivation follows:

$$
\begin{aligned}
||\boldsymbol{W}(D') - \boldsymbol{W}(D)|| &= \sum_{C_i} (g(D', C_i) - g(D, C_i)) \\
&= \sum_{C_i} ((\sum_{T \in D} \cdot + \sum_{T \in \{\tau\}} \cdot) - \sum_{T \in D} \cdot) \\
&= \sum_{C_i} \sum_{T \in \{\tau\}} \frac{\#\text{ of occurrences of } C_i \text{ in } T}{|T|} \\
&= \sum_{T \in \{\tau\}} \sum_{C_i} \frac{\#\text{ of occurrences of } C_i \text{ in } T}{|T|} \\
&= \sum_{T \in \{\tau\}} \frac{1}{|T|} \sum_{C_i} (\#\text{ of occurrences of } C_i \text{ in } T) \\
&= \sum_{T \in \{\tau\}} \frac{|T|}{|T|} = 1
\end{aligned}
$$

In the definition of g, we had normalized cell visits via dividing them by $|T|$. The intuition behind this can be observed from the above derivation. Without normalization, the last step reduces to $\sum_{T \in \{\tau\}} |T|$. In the general case this quantity cannot be bounded, leading to unbounded sensitivity ΔW. Previous works facing this problem resort to: (i) truncation, i.e., keep only the first T_{max} points of trajectories and remove the remaining points (where T_{max} is an input parameter), or (ii) random sampling, i.e., randomly choose T_{max} points from each trajectory and remove the rest [16, 49].

Both approaches yield $\Delta W = T_{max}$. However, we argue that our normalization approach is more appropriate for our density measurement goal. The problem with truncating is that by removing all points after T_{max}, we lose significant portions of trajectories that are much longer than T_{max}. This can be circumvented by having a high T_{max}, but doing so increases sensitivity, beating the purpose of bounding ΔW in the first place. The problem with sampling is that a sample will randomly throw away many locations that could significantly contribute to spatial densities. Our approach ensures that all entries are properly considered in density calculation.

B INTUITION AND WORKED EXAMPLE FOR ORDINARY LEAST SQUARES

In Section 4.3, we use Ordinary Least Squares (OLS) for post-processing and optimizing noisy trip counts $\hat{h}$. We intuitively explain the benefit of OLS using the following example.

Let C_1 and C_2 be two grid cells in the top-level of AdaTrace's adaptive grid. Let $M_1 = 2$ and $M_2 = 3$, i.e., in the bottom-level, C_1 is divided into 4 cells (denoted $C_{1,1}, C_{1,2}, C_{1,3}, C_{1,4}$) and C_2 is divided into 9 cells (denoted $C_{2,1} \dots C_{2,9}$). Without noise, we trivially have: $h(C_1 \rightsquigarrow C_2) = \sum_{k=1}^{4} \sum_{l=1}^{9} h(C_{1,k} \rightsquigarrow C_{2,l})$. That is, the total is consistent, and equals the sum of its parts. For example, observe the trip counts in Figure 3: $h(C_1 \rightsquigarrow C_1) = 1$ at the top-level (LHS), and $h(C_{1,1} \rightsquigarrow C_{1,3}) = 1$ at the bottom-level (RHS). However, with random Laplace noise, consistency is no longer guaranteed, and for noisy counts we may have: $\hat{h}(C_1 \rightsquigarrow C_2) \neq \sum_{k=1}^{4} \sum_{l=1}^{9} \hat{h}(C_{1,k} \rightsquigarrow C_{2,l})$.

Our goal in using OLS is to re-establish consistency by post-processing $\hat{h}$ and obtaining $\hat{h}'$, which denotes a consistent and optimized trip count. $\hat{h}'$ values are then used in the definition of the trip distribution $\mathcal{R}$, in place of $\hat{h}$. We establish consistency by defining $\hat{h}'(C_1 \rightsquigarrow C_2)$ as a linear combination of $\hat{h}(C_1 \rightsquigarrow C_2)$ and $\sum_{k=1}^{4} \sum_{l=1}^{9} \hat{h}(C_{1,k} \rightsquigarrow C_{2,l})$. A natural first attempt towards linear combination is averaging:

$$\hat{h}'(C_1 \rightsquigarrow C_2) = \frac{1}{2}\hat{h}(C_1 \rightsquigarrow C_2) + \frac{1}{2}\sum_{k=1}^{4}\sum_{l=1}^{9}\hat{h}(C_{1,k} \rightsquigarrow C_{2,l})$$

However, observe that the noise variance in this case is $Var(\hat{h}') = {}^1/_4 Var(\hat{h}) + 36({}^1/_4)Var(\hat{h}) = {}^{37}/_4 Var(\hat{h})$, significantly higher than $Var(\hat{h})$, which means using the original value $\hat{h}$ is better than computing $\hat{h}'$. This is caused by two factors. First, averaging does not take into account the possibly uneven privacy budget assigned when retrieving top-level counts versus bottom-level counts. (Recall from Section 4.3 that top-level counts are retrieved with budget $\theta\varepsilon_3$, whereas bottom-level counts are retrieved using budget $(1-\theta)\varepsilon_3$.) Second, averaging does not take into account the difference in the number of times noise is added to each of the two parts – according to the properties of variance, the sum of 36 i.i.d. Laplace random variables has higher variance than 1.

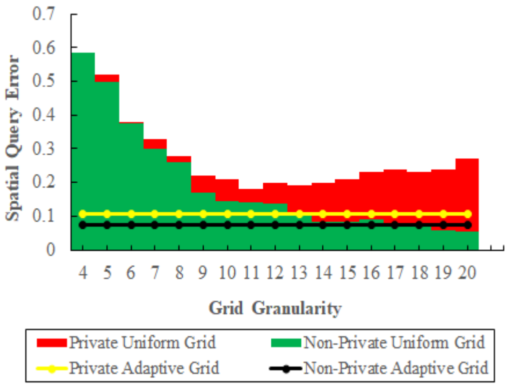

Figure 8: Comparing private and non-private versions of uniform grids versus our density-aware adaptive grid. Results are obtained using the Brinkhoff dataset and $\varepsilon = 1.0$. Grid granularity (x axis) controls the granularity of the uniform grid, e.g., a value of 10 means the uniform grid is 10x10.

The following OLS approach addresses these shortcomings and calculates $\hat{h}'$ as follows:

$$\hat{h}'(C_1 \rightsquigarrow C_2) = \frac{36\theta^2}{(1-\theta)^2 + 36\theta^2} \cdot \hat{h}(C_1 \rightsquigarrow C_2) + \frac{(1-\theta)^2}{(1-\theta)^2 + 36\theta^2} \cdot \sum_{k=1}^{4}\sum_{l=1}^{9}\hat{h}(C_{1,k} \rightsquigarrow C_{2,l})$$

It can be verified that $Var(\hat{h}'(C_1 \rightsquigarrow C_2)) < Var(\hat{h}(C_1 \rightsquigarrow C_2))$, which improves accuracy. This computation is bottom-up in nature, since top-level count is optimized using bottom-level counts. As a final step, consistency requires that the sum of the bottom-level counts equals the top-level count. To achieve this, we employ a top-down approach by distributing the acquired difference $\delta_{OLS} = \hat{h}'(C_1 \rightsquigarrow C_2) - \sum_{m=1}^{4}\sum_{n=1}^{9} \hat{h}(C_{1,m} \rightsquigarrow C_{2,n})$ equally among all bottom-level counts. The resulting post-processed bottom level counts in our example become:

$$\hat{h}'(C_{1,k} \rightsquigarrow C_{2,l}) = \hat{h}(C_{1,k} \rightsquigarrow C_{2,l}) + \frac{\delta_{OLS}}{36}$$

C USEFULNESS OF DENSITY-ADAPTIVE GRID

To illustrate the usefulness of our adaptive grid, we compare it with a uniform grid with and without privacy constraints. We run an experiment similar to the spatial query experiment in Section 6 using D_{real} = Brinkhoff. We first find how many times each cell in Brinkhoff's grid is visited. Then, we issue random counting queries of the form: *How many times was region R visited?*, where R typically spans portions of multiple cells. Queries are answered assuming locations are uniformly distributed within each cell, e.g., if query region R contains $^1/_2$ of C_1 and $^1/_3$ of C_2, and C_1 is visited 10 times and C_2 is visited 12 times, the query answer would be computed as $5 + 4 = 9$. We find the relative error in the query answer by comparing this computed answer with the actual answer (i.e., if the query was answered using D_{real}). While the above procedure is followed for non-private grids, for private grids, an additional step is taken before the query answers are computed: Appropriate

Laplace noise is added to the visit count of each cell, and queries are answered using noisy counts instead of actual counts.

We give the results of the above experiment in Figure 8. We make three observations. First, by analyzing the private and non-private versions of the uniform grid, we observe the two sources of error in $\Omega(D)$ discretization. When grid granularity is too low, the dominating error is the coarse granularity. This is confirmed by the observation that increasing the grid granularity will almost always decrease error in the non-private case. However, in the private case, after the grid granularity exceeds 14x14, query error starts increasing. This is caused by the Laplace noise, which explains the inverse bell-shaped readings for the private uniform grid. Second, we observe that the non-private adaptive grid is as good as a high-granularity uniform grid (e.g., 16x16). The uniform grid performs better only after its granularity exceeds 19x19. Thus, the adaptive grid is useful and efficient even when no noise is present. Third, upon observing that non-private and private versions of the adaptive grid perform similarly, we can conclude that our process of building the grid is not too negatively impacted by Laplace noise, as opposed to the uniform grid where Laplace noise destroys utility when grid granularity is high.

D ADDITIONAL EXPERIMENTS

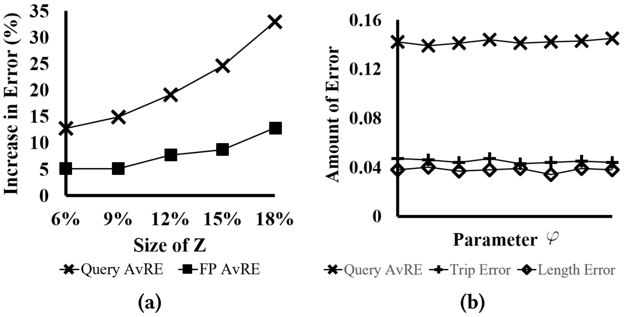

Figure 9: (a) The size of sensitive zone Z is positively correlated with error amounts. (b) φ has no observable impact on aggregate utility, when using the Brinkhoff dataset.